

1/5/1

DIALOG(R)File 351:Derwent WPI

(c) 2003 Thomson Derwent. All rts. reserv.

013196222 **Image available**

WPI Acc No: 2000-368095/200032

XRPX Acc No: N00-275535

Information processing system for high security IC card has CPU that executes given process in accordance with program to process data, which includes one or more data process devices

Patent Assignee: HITACHI LTD (HITA)

Inventor: FUKUZAWA Y; KAMINAGA M; OKI M; OKUHARA S; OHKI M

Number of Countries: 030 Number of Patents: 006

Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
EP 1006492	A1	20000607	EP 99123518	A	19991125	200032 B
JP 2000165375	A	20000616	JP 98338779	A	19981130	200036
CN 1255692	A	20000607	CN 99125831	A	19991130	200046
KR 2000052392	A	20000825	KR 9953692	A	19991130	200121
US 6408075	B1	20020618	US 99449537	A	19991129	200244
			US 2000525014	A	20000314	
TW 466393	A	20011201	TW 99120917	A	19991201	200252

Priority Applications (No Type Date): JP 98338779 A 19981130

Patent Details:

Patent No	Kind	Lan Pg	Main IPC	Filing Notes
-----------	------	--------	----------	--------------

EP 1006492	A1	E	42	G07F-007/10
------------	----	---	----	-------------

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT
LI LT LU LV MC MK NL PT RO SE SI

JP 2000165375	A	28	H04L-009/10
---------------	---	----	-------------

CN 1255692	A		G06K-019/07
------------	---	--	-------------

KR 2000052392	A		G06K-019/07
---------------	---	--	-------------

US 6408075	B1		H04L-009/28	Div ex application US 99449537
------------	----	--	-------------	--------------------------------

TW 466393	A		G06C-001/00
-----------	---	--	-------------

Abstract (Basic): EP 1006492 A1

NOVELTY - A CPU (201) executes a given process in accordance with a program to process a data. The program includes one or more data process devices, each of which has a process instruction for giving an execution instruction to the CPU. The data is a combination of a normal data and its bit inverted data, which include data under a normal process and data obtained by inverting bits of the data under the normal process.

USE - In IC card - smart card.

ADVANTAGE - Provides tamper resistant device of high security. Reduced correlation or dependency between data processing and its consumption current of an IC chip.

DESCRIPTION OF DRAWING(S) - The drawing shows a hardware structure of an IC card chip.

CPU (201)

pp; 42 DwgNo 1,2/31

Title Terms: INFORMATION; PROCESS; SYSTEM; HIGH; SECURE; IC; CARD; CPU;

EXECUTE; PROCESS; ACCORD; PROGRAM; PROCESS; DATA; ONE; MORE; DATA;

PROCESS; DEVICE

Derwent Class: T01; T04; T05; U21

International Patent Class (Main): G06C-001/00; G06K-019/07; G07F-007/10;

H04L-009/10; H04L-009/28

International Patent Class (Additional): G06F-001/00; G06F-007/552;

G06K-019/073; G09C-001/00

File Segment: EPI

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号
特開2000-165375
(P2000-165375A)

(43) 公開日 平成12年6月16日 (2000.6.16)

(51) Int.Cl. ⁷	識別記号	FI	テーマコード (参考)
H04L 9/10		H04L 9/00	621Z 5B035
G06K 19/073		G09C 1/00	660A 5J104
G09C 1/00	660	G06K 19/00	P

審査請求 未請求 請求項の数34 OL (全 28 頁)

(21) 出願番号 特願平10-338779

(22) 出願日 平成10年11月30日 (1998.11.30)

(71) 出願人 000005108

株式会社日立製作所
東京都千代田区神田駿河台四丁目6番地

(72) 発明者 大木 優

東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内

(72) 発明者 福澤 寧子

神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

(74) 代理人 100068504

弁理士 小川 勝男

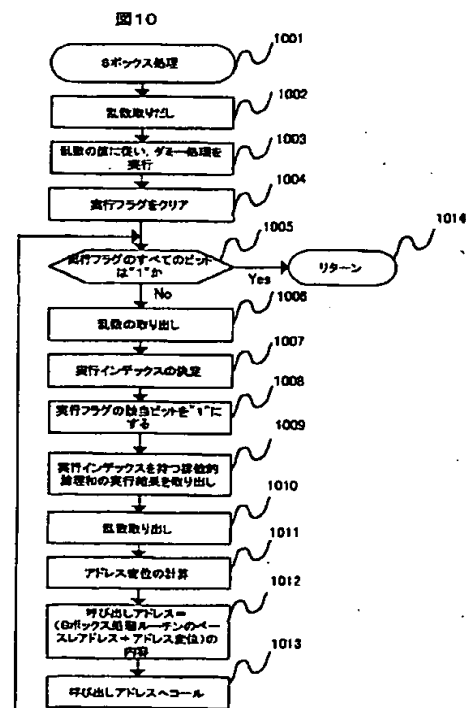
最終頁に続く

(54) 【発明の名称】 情報処理装置、ICカード

(57) 【要約】

【課題】 ICカード用チップでの処理のデータとICカード用チップの消費電流との関連性を減らす。

【解決手段】 ICカード用チップの処理順序をランダムに入れ替えたり、無駄なダミー処理を追加することにより、処理データとICカード用チップの消費電流との関連性を減らす。ICカード用チップでの処理のデータとICカード用チップの消費電流との関連性を減らすことを可能とする。



【特許請求の範囲】

【請求項 1】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い所定の処理を実行しデータ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理のデータとそのビットの値を反転したデータとからなる正常データ及びビット反転データの組みをデータとして有することを特徴とする情報処理装置。

【請求項 2】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、同一の命令で、正常処理のデータとそのビットの値を反転したデータとを処理する正常データ及びビット反転データ処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 3】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理命令と同じ命令コマンドを有し、かつ正常処理命令において処理されるデータのビットの値を反転したデータを処理する命令を有する正常データ命令及びビット反転データ命令処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 4】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、正常処理の結果のビットの値を反転したデータを結果として生成するビット反転データ生成手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 5】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段として、異なる処理データを複数繰り返して処理する繰り返しデータ処理手段を含む、情報処理装置において、繰り返しデータ処理手段をランダムに実行する繰り返しランダム実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 6】プログラムを格納するプログラム格納部と、データを保存するデータ格納部を持つ記憶装置と、プログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置において、プログラム内にプログラムの該当処理に影響を与えないダミー実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

10 【請求項 7】請求項 5 または 6 に記載の情報処理装置において、ダミー実行処理手段と繰り返しデータ処理手段とをランダムに実行するダミー実行及び繰り返しランダム実行処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 8】請求項 1 から 7 に記載の情報処理装置において、ビット単位のデータを入れ替える転置処理手段をデータ処理手段として有することを特徴とする情報処理装置。

20 【請求項 9】請求項 1 から 7 に記載の情報処理装置において、バイト単位のデータを入れ替える換字処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 10】請求項 1 から 7 に記載の情報処理装置において、入力データと暗号鍵データの排他的論理和を行う排他的論理和処理手段と入力データのビットを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他的論理和処理手段をデータ処理手段として有することを特徴とする情報処理装置。

30 【請求項 11】請求項 1 から 7 に記載の情報処理装置において、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と、入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段とをデータ処理手段として有することを特徴とする情報処理装置。

40 【請求項 12】請求項 1 から 7 に記載の情報処理装置において、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段と、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段をデータ処理手段として有することを特徴とする情報処理装置。

50 【請求項 13】請求項 1 から 7 に記載の情報処理装置において、入力データと暗号鍵データの排他的論理和を行う排他的論理和処理手段と、入力データを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他

的論理処理手段、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 14】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、入力データと暗号鍵を使って、べき乗乗算を繰り返してべき乗乗算計算を行う手段を有するデータ処理手段を有する情報処理装置において、暗号鍵のビットの値に関わらず、入力データとこれまで計算した結果を乗算剰余する乗算剰余処理手段と、暗号鍵のビットが 1 ならば、乗算剰余処理手段の結果を使い、0 であれば、乗算剰余処理手段の結果を無視する乗算剰余処理結果選択処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 15】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、複数ビットの値に対応して入力データのべき乗剰余計算した結果であるビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、暗号鍵の複数ビットの値毎に、ビット値対応入力データべき乗剰余計算処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段、をもつ入力データと暗号鍵を使ってべき乗剰余計算を行うデータ処理手段を有する情報処理装置において、ビット対応乗算剰余計算処理手段を繰り返し処理しているあるタイミングにおいて、ビット値対応入力データべき乗剰余値を変更するビット値対応入力データべき乗剰余値変更処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 16】請求項 15 に記載の情報処理装置におい

て、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更しない値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項 17】請求項 15 に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項 18】請求項 17 に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると 1 になる二つの値 v と u の中の一方向である v 、あるいは v のべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段において、 v を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に u を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段をビット値対応入力データべき乗剰余値変更部とビット値対応入力データべき乗剰余値変更処理逆変換処理手段として有することを特徴とする情報処理装置。

【請求項 19】請求項 18 に記載の情報処理装置において、上記ビット値対応入力データべき乗剰余値変更処理手段は、剰余演算の法 N において掛けると 1 になる二つの値として、 2 と $N+1/2$ を使うことを特徴とする情報処理装置。

【請求項 20】請求項 18 に記載の情報処理装置において、暗号鍵を M ビットの組として表し、そのビットが 0 あるいは 1 であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると 1 になる二つの値 v と u の中の一方向である v 、あるいは v のべき乗を該当剰余演算の法 N において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、暗号鍵の複数ビット M の値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段、 v を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に u を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項 21】請求項 20 に記載の情報処理装置において、あるタイミングで、ビット値対応入力データべき乗剰余値に、 v として 2 のランダムなべき乗を該当剰余演

10

20

30

40

50

算の法Nにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、2を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に $(n+1)/2$ を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項22】請求項17に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、2の任意数のべき乗、あるいは2の8乗の任意数のべき乗を剰余演算の法に掛けた数を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項23】プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、剰余演算を行うデータ処理手段を有する情報処理装置において、剰余演算の法を変更する剰余演算法変更処理手段と、剰余演算の右辺の値を変更する剰余演算右辺変更処理手段、剰余演算法変更処理手段で変更した剰余演算の法を変更前の計算結果に戻す剰余演算法変更処理逆変換処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項24】請求項23の剰余演算法変更処理手段と剰余演算右辺変更処理手段、剰余演算法変更処理逆変換処理手段において、法の変更手段として、整数倍を掛ける剰余演算法変更処理手段と、剰余演算の結果を剰余演算法変更処理手段で使用した整数を剰余演算の右辺の値に掛ける剰余演算右辺変更処理手段、計算結果に戻す手段として、剰余演算の結果を剰余演算法変更処理手段で使用した整数でわる剰余演算法変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項25】請求項24の剰余演算法変更処理手段と剰余演算右辺変更処理手段、剰余演算法変更処理逆変換処理手段において、整数倍として2の任意数のべき乗、あるいは、2の8乗の任意数のべき乗を使用する剰余演算法変更処理手段と、剰余演算の右辺の値を、剰余演算法変更処理手段で使用した整数倍として2の任意数のべき乗、あるいは、2の8乗の任意数のべき乗で掛ける剰余演算右辺変更処理手段、計算結果に戻す手段として、剰余演算の結果を剰余演算法変更処理手段で使用した整数倍として2の任意数のべき乗、あるいは、2の8乗の任意数のべき乗でわる剰余演算法変更処理逆変換処理手段、をデータ処理手段として有することを特徴とする情報処理装置。

【請求項26】プログラムを格納するプログラム格納

部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなり、データ処理手段として、剰余演算を行うデータ処理手段を有する情報処理装置において、ビット値対応入力データべき乗剰余値を変更するビット値対応入力データべき乗剰余値変更処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項27】請求項26に記載の情報処理装置において、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更しない値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理手段をデータ処理手段として有することを特徴とする情報処理装置。

【請求項28】請求項26に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項29】請求項28に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると1になる二つの値 v と u の中の一つである v 、あるいは v のべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段 v を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に u を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段をビット値対応入力データべき乗剰余値変更部とビット値対応入力データべき乗剰余値変更処理逆変換処理手段として有することを特徴とする情報処理装置。

【請求項30】請求項29に記載の情報処理装置において、上記ビット値対応入力データべき乗剰余値変更処理手段において剰余演算の法Nにおいて掛けると1になる二つの値として、2と $N+1/2$ を使うことを特徴とする情報処理装置。

【請求項31】請求項29に記載の情報処理装置において、暗号鍵をMビットの組として表し、そのビットが0あるいは1であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると1になる二つの値 v と u の中の一つである v 、あるいは v のべき乗を該当剰余演算の法Nにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手

段、暗号鍵の複数ビットMの値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段vを剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果にuを剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項32】請求項31に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、vとして2のランダムなべき乗を該当剰余演算の法Nにおいて掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、2を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に(n+1)/2を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を有することを特徴とする情報処理装置。

【請求項33】請求項26に記載の情報処理装置において、ビット値対応入力データべき乗剰余値に、2の任意数のべき乗、あるいは2の8乗の任意数のべき乗を剰余演算の法に掛けた数を加えるビット値対応入力データべき乗剰余値変更処理手段をビット値対応入力データべき乗剰余値変更処理手段として有することを特徴とする情報処理装置。

【請求項34】上記情報処理装置は、ICカードであることを特徴とする請求項1から33に記載の情報処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、情報処理装置、特に、機密性の高いICカードなどの耐タンパ装置に関するものである。

【0002】

【従来の技術】ICカードは、主に、勝手に書き換えられない情報の保持や秘密情報である暗号鍵を使ったデータの暗号化や暗号文の復号化を行うために使われる装置である。ICカードは、電源を持っていないため、リーダライタに差し込まれると、電源の供給を受け、動作可能となる。動作可能になると、リーダライタからコマンドを受けて、コマンドに従い、データの転送を行う。ICカードの一般的な解説は、オーム社出版電子情報通信学会編水沢順一著「ICカード」などにある。

【0003】ICカードの構成は、図1に示すように、カード101の上に、ICカード用チップ102を搭載したものである。一般にICカードは、接点を持ち、接点を通して、リーダライタから電源の供給やリーダライタとのデータの通信を行う。

【0004】ICカード用チップの構成は、基本的にマイクロコンピュータと同じような構成である。その構成は、図2に示すように、中央演算装置201、記憶装置

204、入出力ポート207、コ・プロセッサ202からなる。中央処理装置201は、論理演算や算術演算などを行う装置であり、記憶装置204は、プログラムやデータを格納する装置である。入出力ポートは、リーダライタと通信を行う装置である。コ・プロセッサは、剰余演算を行うための特別な演算装置であり、非対称暗号であるRSAの演算などに用いられる装置である。ICカード用プロセッサの中には、コ・プロセッサを持たないものも多くある。データバス203は、各装置を接続するバスである。

【0005】記憶装置204は、ROM(Read Only Memory)やRAM(Random Access Memory)、EEPROM(Electrical Erasable Programmable Read Only Memory)などからなる。ROMは、変更できないメモリであり、主にプログラムを格納するメモリである。RAMは自由に書き換えができるメモリであるが、電源の供給が中断されると、記憶している内容が消えてなくなる。ICカードがリーダライタから抜かれると電源の供給が中断されるため、RAMの内容は、保持できなくなる。EEPROMは、電源の供給が中断されてもその内容を保持することができるメモリである。書き換える必要があり、ICカードがリーダライタから抜かれても、保持するデータを格納するために使われる。例えば、プリペイドカードでのプリペイドの度数などは、使用するたびに書き換えられ、かつリーダライタが抜かれてもデータを保持する必要があるため、EEPROMで保持される。

【0006】ICカードは、プログラムや重要な情報がICカード用チップの中に密閉されているため、重要な情報を格納したり、カードの中で暗号処理を行うために、使われている。ICカードでの暗号処理の解読の難しさは、暗号アルゴリズムの解読の困難さと同じと考えられていた。しかし、ICカードが暗号処理を行っている時の消費電流を観測し、解析することにより、暗号アルゴリズムの解読より容易に暗号処理の内容や暗号鍵が推定される可能性が示唆されている。消費電流は、リーダライタから供給されている電流を測定することにより観測することができる。これは、John Wiley & sons社 W. Rankl & W. Effing著「Smart Card Handbook」の8.5.1.1 Passive protective mechanisms(263ページ)にこのような危険性が記載されている。

【0007】ICカード用チップを構成しているCMOSは、出力状態が1から0あるいは0から1に変わった時に電流を消費する。特に、データバス203のバスは、大きな電気容量を持つため、バスの値が1から0あるいは0から1に変わると、大きな電流が流れる。そのため、消費電流を観測すれば、ICカード用チップの中で、何が動作しているか分かる可能性を示唆している。

【0008】図5は、ICカード用チップの1サイクルでの消費電流の波形を示したものである。処理しているデータの依存して、電流波形が501や502のように異

なる。このような差は、バス 203 を流れるデータや中央演算装置 201 で処理しているデータに依存して生じる。

【0009】コ・プロセッサ 202 は、中央演算装置と並列に、長いビット、例えば、512 ビットの剰余演算を行うことができる。そのため、中央演算装置の消費電流とは異なった消費電流波形を長い時間、観測することができる。その特徴的な波形を観測すれば、コ・プロセッサの動作回数を容易に測定することができる。コ・プロセッサの動作回数が暗号鍵と何らかの関係があるならば、コ・プロセッサの動作回数から暗号鍵を推定できる可能性がある。

【0010】また、コ・プロセッサでの演算内容が、暗号鍵に依存した偏りがあると、その偏りが消費電流から求められ、暗号鍵が推定される可能性がある。

【0011】中央演算装置でも同様である。暗号鍵のビットの値は決まっているため、処理するデータを変更して、消費電流を観測することにより、暗号鍵のビットの値の影響が観測できる可能性がある。これらの消費電流の波形を統計的に処理することにより、暗号鍵を推定できる可能性がある。

【0012】

【発明が解決しようとする課題】本発明の課題は、IC カード用チップでのデータ処理と消費電流との関連性を減らすことである。消費電流とチップの処理との関連性が減れば、観測した消費電流の波形から IC カードチップ内での処理や暗号鍵の推測が困難になる。本発明の着眼点は、IC カードチップでの処理手順をランダムにすることや、ダミー処理を挿入することにより、消費電流の波形から、処理や暗号鍵の推測を困難にするものである。

【0013】

【課題を解決するための手段】IC カード用チップに代表される耐タンパ装置は、プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置とプログラムに従い、所定の処理を実行し、データ処理を行う中央演算装置を持ち、プログラムは、中央演算装置に実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる、情報処理装置として考えることができる。本発明では、処理しているデータと IC カード用チップの消費電流の関連性を減らすための方法の一つとして、処理するデータを正常処理のデータとそのビットの値を反転したデータの両方を使用する。そして、ビット反転のデータと正常データを、同一の命令で、正常処理のデータとビットの値を反転したデータとを処理することにより、データバス上の値などが“0”から“1”、あるいは“1”から“0”なる回数を一定にする。データバスでは、値が反転したときに電流を大きく消費するため、“1”と“0”の反転回数が一定になることにより、電流の消費する回数が一定になり、データ処理と消費電流の関連性を減らすことができる。

【0014】正常データとビット反転データを同じように処理するための別の方法として、同一ルーチンで正常データとビット反転データを処理できない場合は、正常処理命令と同じ命令コマンドを有し、ビット反転したデータを処理する命令を有する処理ルーチンを使用する。さらに、常に処理した結果でも正常データとビット反転データを生成することを行うようにし、正常データとビット反転データを常に同じように処理するようにする。

【0015】データ処理と消費電流の関連性を減らす別の方法は、処理するデータだけ異なる繰り返し処理があれば、一定の順番でデータを取り出し処理するのではなく、処理順番をランダムにする方法である。また、別の方法は、プログラムの該当処理に影響を与えないダミー実行処理を入れることにより、どこで何をしているかが一定にならないようにする方法である。ダミー実行処理手段と繰り返しデータ処理手段とをランダムに実行することを組み合わせることは、さらに、データ処理と消費電流の関連性を減らすために有効である。

【0016】正常データとビット反転データを使う方法や、ダミー実行処理手段と繰り返しデータ処理手段を使う方法は、暗号アルゴリズムで使われるビット単位のデータを入れ替える転置処理手段やバイト単位のデータを入れ替える換字処理手段のデータ処理と消費電流の関連性を減らすために、特に有効である。

【0017】実際、DES などの転置や換字を使う暗号では、排他的論理和を使うことが多いので、入力データと暗号鍵データの排他的論理和を行う排他的論理処理手段と入力データのビットを反転したデータと暗号鍵の排他的論理和を行うビット反転データ排他的論理処理手段を有することは、データ処理と消費電流の関連性を減らすために、有効である。また、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と入力データのビットを反転したデータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。そして、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段と入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。

【0018】それらを組み合わせて、入力データに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段と、入力データのビットを反転したデ

ータに対して非線形の換字を行い、換字した結果及び換字した結果のビットの値を反転したビット反転データを生成する非線形換字処理手段、入力データに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段、入力データのビットを反転したデータに対して非線形の転置を行い、転置した結果および転置した結果のビットの値を反転したビット反転データを生成する非線形転置処理手段とを有することは、データ処理と消費電流の関連性を減らすために、有効である。

【0019】素因数分解が困難であることを利用したRSAなどの暗号では、入力データと暗号鍵を使って、べき乗乗算を繰り返してべき乗剰余計算を行う。データ処理と消費電流の関連性を減らすための方法の一つは、暗号鍵のビットの値に関わらず、入力データとこれまで計算した結果を乗算剰余する乗算剰余処理手段と、暗号鍵のビットが1ならば、乗算剰余処理手段の結果を使い、0であれば、乗算剰余処理手段の結果を無視する乗算剰余処理結果選択処理手段を使用することにより、暗号鍵のビットの値に関わらず、乗算剰余を行うことである。このようにすることにより、乗算剰余のデータ処理と乗算剰余の消費電流の関連性を減り、乗算剰余の実行回数から、暗号鍵を推定することは困難となる。

【0020】RSAでは、複数ビットの値に対応して入力データのべき乗剰余計算した結果であるビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、暗号鍵の複数ビットの値毎に、ビット値対応入力データべき乗剰余計算処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段を使い、性能を改善する方法がある。しかし、この方法では、常に、同じビット値対応入力データべき乗剰余値の組を使用するため、データと消費電流に関連性から暗号鍵を推定される可能性がある。これを解決する方法一つは、ビット対応乗算剰余計算処理手段を繰り返し処理しているあるタイミングにおいて、ビット値対応入力データべき乗剰余値変更するビット値対応入力データべき乗剰余値変更処理手段を用い、ビット値対応入力データべき乗剰余値を一定の回数で変更する方法である。また、この方法では、変更する方法に依存するが、ビット値対応入力データべき乗剰余値の変更を元に戻す必要がある場合がある。その一つの方法は、ビット値対応入力データべき乗剰余値変更処理手段でビット値対応入力データべき乗剰余値に変更を加えた処理結果を変更を加えなかった値に戻すビット値対応入力データべき乗剰余値変更処理逆変換処理を行う方法である。ビット値対応入力データべき乗剰余値変更処理手段には、いくつかの方法があるが、ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えることも一つの方法である。別な方法は、ビット値対応入力データべき乗

剰余値に、剰余演算の法において掛けると1になる二つの値 v と u の中の一つである v 、あるいは v のべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段とビット値対応入力データべき乗剰余値変更処理手段において、 v を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に u を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を用いる方法である。また、ビット値対応入力データべき乗剰余値変更処理手段で使う、剰余演算の法 N において掛けると1になる二つの値として、 2 と $N+1/2$ を使うことは、 v と u を簡単に求める方法の一つである。RSAのような暗号を処理するために、暗号鍵を M ビットの組として表し、そのビットが0あるいは1であるすべての組み合わせに対して、その値を、入力データにべき乗乗算してビット値対応入力データべき乗剰余値を求めるビット値対応入力データべき乗剰余計算処理手段と、あるタイミングで、ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると1になる二つの値 v と u の中の一つである v 、あるいは v のべき乗を該当剰余演算の法 N において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、暗号鍵の複数ビット M の値毎に、ビット値対応入力データべき乗剰余値変更処理手段で計算したビット値対応入力データべき乗剰余値とこれまで乗算剰余した結果を乗算剰余するビット対応乗算剰余計算処理手段、 v を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に u を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を使うことは、データと処理と消費電流の関連性を減らすために有効である。また、その時、あるタイミングで、ビット値対応入力データべき乗剰余値に、 v として2のランダムなべき乗を該当剰余演算の法 N において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段、2を剰余演算の法として掛けた回数だけ、べき乗剰余の処理結果に $(n+1)/2$ を剰余演算の法として掛ける処理を行うビット値対応入力データべき乗剰余値変更処理逆変換処理手段を使うと、 v と u を簡単に求めることができる。

【0021】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。

【0022】図1はICカードの外観を示したものである。ICカード101は、ISO7816の規格により大きさや、ICカードのチップ102の位置や接点の数および割り当てなどが規定されている。

【0023】図2は、ICカード用チップ102の内部構成である。構成については、従来技術の説明で既に述べたとおりである。本発明は、プログラム205をいつも一定に実行させるのではなく、ランダムに動作させたり、無駄なダミー処理を追加したりして、ICカード用チ

ップのハードウェアが消費する電流波形の再現を困難にするものである。

【0024】本実施例では、対称暗号DES(data encryption standard)と非対称暗号RSAを例にする。これは、他の暗号にも使うことができる。DESやRSAなどの暗号については、共立出版株式会社岡本栄司著「暗号理論入門」などに、記載されている。

【0025】DESは、64ビットのデータ(平文か暗号文)を56ビットの暗号鍵で、暗号化と復号化を行う。暗号化と復号化で同じ暗号鍵を使用するため、対称暗号と呼ばれている。DESは、トランプをきって、ランダムにするように、平分(暗号化される文)のビットをランダムに入れ替え暗号化する。データの入れ替えは、暗号鍵にしたがって行う。復号化する時は、暗号鍵に従い入れ替えた順と逆に入れ替えを行い、データを元に戻す。DESの処理での入れ替えは、ビット単位と複数ビットまとまった単位での二つの入れ替え方法がある。前者は、転置と呼び、後者は、換字と呼ぶ。

【0026】図3を用いて、DESの暗号化の方法を説明する。暗号文は、まず、初期転置(IP: Initial Permutation)301で転置させられる。これは、初期転置テーブルに従い、ビット単位で、暗号文の64ビットのデータを入れ替える。これ以降、初期転置の逆転置(IP-1)313まで、一組の操作を16段行う。

【0027】各1段の処理は、前段の結果の前半か後半の32ビットのデータと暗号鍵を入力としてf関数303と呼ばれる処理を行い、その出力を前段の残りの半分のビットを使って排他的論理和305を取る操作を行う。暗号鍵も、入れ替えが行われる。暗号鍵に対して、まず、PC-1というテーブルを使った選択転置PC-1(302)が行われる。その後、PC-2というテーブルを使った選択転置PC-2 304を行い、暗号鍵の入れ替えを行う。次の段では、28ビットづつをRSテーブルにしたがって巡回させて使用する。

【0028】f関数303の処理を、図4に示す。まず、f関数への入力文を選択的転置行列Eに基づいて、選択的転置を行う(402)。次に、選択的転置を行った入力データと暗号鍵との排他的論理和をとり(403)、Sボックスの処理を行い(404)、P転置処理を行う(405)。Sボックスの処理は、403の排他的論理和の処理結果である48ビットから6ビットづつ取り出し、Sボックステーブルの行番号と列番号を求め、4ビットのデータを生成する処理である。各Sボックステーブルは、6ビットごとのデータの位置により異なる。P転置処理は、P転置テーブルに従い、32ビットのビット位置を入れ替える操作である。

【0029】処理手順が同一であれば、データに依存して、消費電流波形の偏りが出るため、本発明では、図4の各処理を、実行ごとに処理手順を変更する、ランダムな処理を追加することを行う。

【0030】図24を用いて、選択的転置処理402について説明する。選択的転置とは、f関数に入力される入力データを選択的転置行列Eによって、ビットの位置の置き換えと入力データを32ビットから48ビットに拡大するものである。選択的転置行列E(2701)を図27に示す。左上(2702)から右下(2703)に数えて、入力データの第32、1、2、...、32、1番目のビットを出力の第1番目から第48番目のビットにする。

【0031】選択的転置処理の手順を説明する前に、本実施例における選択的データ転置で使用するデータについて説明する。f関数に入力されるデータは、図6に示すように、前の処理で正常データ602とそのビット603を反転したビット反転データの組にして入力される。このような組で扱うと、常に、“0”のビットの数と“1”のビットの数は一定になる。例えば、正常データのビットが、“00111100”であれば、ビット反転データは“11000011”となり、正常データとそのビット反転データの組での“0”のビットの数は8であり、“1”のビットの数、すなわちハミングウェイトは8である。そのため、正常データとそのビット反転データを組で、データバス(203)に転送したならば、組における“0”と“1”のビットの数が常に一定になるため、データの依存性が消費電流に反映されないという特徴がある。例えば、データバス203がプリチャージバスであれば、データを転送する準備として、一旦データバスの値を“1”か“0”にそろえ、その後、データを転送する。この場合、常に、プリチャージの状態では、バスの値が同じ値になっているため、正常データとそのビット反転データを組でデータ転送した場合、“1”から“0”あるいは、“0”から“1”に値が変わる数は、プリチャージの状態の前後で、一定になり、データによる消費電流依存性が減る。もし、データバスが、プリチャージ状態を持たない、スタティックバスであれば、ビットの反転は、前のデータの値に依存するが、転送するデータのハミングウェイトが常に一定のため、転送するデータによる電流消費依存性は少ない。少なくとも、正常データだけでは、ハミングウェイトは一定でないため、その電流消費依存性が生じる。正常データとそのビット反転データの組を一緒にデータ転送し、ハミングウェイトをそろえると、転送するデータによる電流消費依存性は減る。

【0032】選択的転置処理の手順を図24に示す。

【0033】選択的転置処理は、入力データの32ビットを6ビット単位で処理し、48ビットのデータを生成する。処理されるデータは、正常な入力データとそのビット反転入力データであり、2バイトで実現され、それぞれのバイトに8ビットずつ格納されている。選択的転置処理では、通常、選択的転置行列Eの並びの順に処理する。しかし、それでは、データの処理順序が常に一定

であるため、消費電流の波形から、処理されているデータを推測される可能性がある。そのため、本実施例では、実行順序をランダムにし、かつランダムにダミー処理を追加し、処理順序が一定にならないようにする。処理順序がランダムになることにより、データと消費電流の依存性を減らすことができる。

【0034】本実施例の選択的転置処理では、実行フラグをクリアする(2402)。この実行フラグは、各ビットが繰り返し処理の処理単位で処理が終了しているならば、“1”であり、未処理であれば“0”である。実行フラグのすべてのビットが“1”であるならば、処理

$$\text{実行インデックス} = (\text{乱数 and } 0x07) \% 8 \quad (\text{式1})$$

となる。乱数を0x07で下位3ビットを取り出し、その値を8で割り、その余りを処理する行番号とする。もし、

$$\text{実行インデックス} = (\text{乱数 and } 0x01F) \% 32 \quad (\text{式2})$$

となる。乱数の下位5ビットを取り出し、その値を32で割り、その余りを処理するビットの番号とする。実行インデックスは、乱数だけから決まるので、順番はランダムになり、かつすでに処理済みのデータを選ぶこともありえる。本実施例では、データ処理と消費電流の波形の依存関係を減らすために、この性質を積極的に利用し

$$K = 1 \text{ を実行インデックス分だけ左にシフトする。} \quad (\text{式3})$$

$$\text{実行フラグ} = \text{実行フラグ XOR } K \quad (\text{式4})$$

式3で、実行インデックスのビット位置のビットを“1”にし、式4で、実行フラグと排他的論理和を取ることで、実行フラグの該当ビットを“1”にする。

【0038】以上の処理により、繰り返し処理の中の処理の単位が乱数によってランダムに選ばれるため、処理の単位の順序が一定にならない。そのため、データ処理と消費電流の依存性がなくなり、処理や暗号鍵の推測を困難にする。さらに、実行する繰り返しの単位は、ランダムに選ばれるため、同じ処理を1度以上処理する可能性がある。これは、処理時間を不定にするため、データ処理と消費電流の依存性を見出すことをさらに難しくする。

【0039】次に、実行インデックスで選ばれた繰り返し処理単位での一つの処理について述べる。これは、選択的転置行列Eを行ごとの処理の一つ行の処理や選択的転置行列Eをビットごとの処理の一つのビットの処理に対応する。まず、実行インデックスに従い、処理すべき正常入力データとビット反転入力データを取り出す(2

$$\text{アドレス変位} = (\text{乱数AND } 0x01) * 2$$

これは、乱数の最下位ビットを取り出し、それを2倍する。ルーチン呼び出すアドレスは、選択的転置ルーチンのアドレスが書かれてあるデータ領域のベースアドレス(2802)にアドレス変位を加える。この例では、ベースアドレスには、正常データを最初に処理するルーチンのアドレス(2802)が書かれ、ベースアドレス+2バイトのアドレスには、ビット反転入力データを最初に処理するルーチンのアドレス(2803)が書かれて

を終了する(2403)。そうでなければ、処理を続け、乱数を取り出す(2404)。乱数は、ICカード用チップが乱数生成装置を持っているならば、その装置から乱数を取り出すのも一つの方法である。それ以外の方法として、ソフト的に擬似乱数を作ることも可能である(共立出版株式会社岡本栄司著「暗号理論入門」61ページから86ページ)。

【0035】繰り返し単位でのどの処理を実行するかを乱数によって実行インデックスを決める(2405)。例えば、選択的転置行列Eを行ごとに処理するのであれば、実行インデックスは、

ビット単位であれば、

ている。

【0036】次に、実行フラグの対応するビットに処理済みを示すために、“1”にする(2406)。一つの手段として、以下の方法がある。

【0037】

407)。本実施例では、正常入力データ以外に反転入力データも同じように選択的転置処理を行う。その時、正常入力データとビット反転入力データがいつも同じ順序で処理されないように、順序をランダム化する。そのために、乱数を取り出し(2408)、アドレス変位を計算する(2409)。そのアドレス変位から呼び出すルーチンのアドレスを求め(2410)、ルーチン呼び出す(2411)。

【0040】アドレス変位を用いてデータ処理ルーチン呼び出す一つの実施例を図28を用いて説明する。メモリ(2801)上に、正常入力データを最初に処理するルーチンのアドレス(2802)とビット反転入力データを最初に処理するルーチンのアドレス(2803)を格納する。二つのアドレスの差がアドレス変位である。アドレスを格納する単位が2バイトとすると、2409のアドレス変位の計算方法の一つは以下になる。

【0041】

$$(\text{式5})$$

いる。乱数の最下位ビットが“0”であれば、アドレス変位は0となり、正常入力データを最初に処理するルーチンのアドレス(2802)が選択され、正常入力データを最初に処理するルーチン(2804)がコールされる。乱数の最下位ビットが“1”であれば、アドレス変位は2となり、ビット反転入力データを最初に処理するルーチンのアドレス(2803)が選択され、ビット反転入力データを最初に処理するルーチン(2804)が

コールされる。

【0042】乱数によって、アドレス変位を計算するため、正常入力データを最初に処理するルーチンかビット反転入力データを最初に処理するルーチンがランダムに選択され、正常入力データとビット反転入力データがいつも同じ順序で処理されないようになる。その結果、正常入力データとビット反転入力データの処理と消費電流波形の依存関係がなくなる。

【0043】正常入力データを最初に処理するルーチンは、図25に示すように、正常入力データの選択的転置を行い(2502)、次に、ビット反転入力データの選択的転置を行う(2503)。ビット反転入力データを最初に処理するルーチンは、図26に示すように、ビット反転入力データの選択的転置を行い(2602)、次に、正常入力データの選択的転置を行う(2603)。実際の選択的転置を行う処理ルーチン(2806)は、それぞれのルーチンからさらに呼び出され、対応するビットの選択的転置が行われる。そして、ビット反転入力データの選択的転置(2503)あるいは(2602)では、正常データを選択的転置を行った結果のビット反転したデータを生成する。これは、選択的転置がビット位置の転置であるため、正常データとビット反転データのビット反転の関係は維持されるためである。正常データとビット反転データの格納方法は、図6で示したように、組みになるように格納する。こうすることにより、データ転送時のデータと消費電流の関連性を少なくすることができる。

【0044】以上の処理により、繰り返し処理すべきデータの一つの処理が終わると、2403に戻る。すべてのデータを処理していなければ、乱数に基づき別の処理すべきデータをランダムに選び、処理を繰り返す。乱数を使ってのみ処理すべきデータを決めているので、同じデータを処理することもある。選択的転置処理では、同じデータに対して2度以上を繰り返しても問題が起きない。

【0045】以上説明したように、本実施例の選択的転置処理において説明した、正常入力データとビット反転入力データは、請求項1の「正常データとそのビット反転データの組」の実施例である。また、図25の正常入力データを最初に処理するルーチンあるいは図26のビット反転入力データを最初に処理するルーチンには、ルーチンの中に、正常データの処理手順とビット反転の処理手順を持っており、請求項3の「正常データ命令及びビット反転データ命令処理手段」の実施例である。また、図25の正常入力データを最初に処理するルーチンあるいは図26のビット反転入力データを最初に処理するルーチンは、正常入力データの選択的転置を行った結果を作成すると同時に、その結果をビット反転した結果も生成する。これは、請求項4の「ビット反転データ生成手段」の実施例の一つでもある。データ処理ルーチン28

06は、選択的転置処理では、実際に、その処理を行うルーチンであり、正常データでもビット反転データでも使用されるルーチンである。これは、請求項2の「正常データ及びビット反転データ処理手段」の一つの実施例である。また、実行フラグと乱数により処理順序をランダムに変える手順(2402から2405)は、請求項5の「繰り返しランダム実行処理手段」の一つの実施例である。実行フラグと乱数によりすでに処理済みのデータもランダムに2度以上処理を行う手順(2402から2406)は、請求項6の「ダミー実行処理手段」の一つの実施例である。そして、実行フラグと乱数を使った処理手順2402から2406は、ダミー処理と繰り返しランダム処理を同時に実現しているため、請求項7の「ダミー実行及び繰り返しランダム実行処理手段」の実施例の一つでもある。図24で述べた選択的転置で述べた処理は、請求項8の典型的な実施例である。

【0046】次に、選択的転置を行った入力データと暗号鍵との排他的論理和処理(403)の実施例について述べる。この処理の基本的な流れは、図24を用いてすでに説明した選択的転置の処理の手順とほとんど同じである。主として異なっている点は、710での呼び出しアドレスの計算が排他的論理和のルーチンのベースアドレスに基づいていることである。排他的論理和処理は、選択的転置処理で作成された、二つのデータを処理する。一つは、正常入力データを選択的転置した正常E転置データであり、もう一つは、ビット反転入力データを選択的転置したビット反転E転置データである。これらは、図6に示した形式で格納し、取り出すときは、正常データとビット反転データの対を一度に取り出す。ビット反転E転置データは正常E転置データのビット反転となっている。正常E転置データと暗号鍵は、ともに48ビットである。また、次の処理であるSボックス処理が6ビット単位の処理であるため、排他的論理和処理では6ビット単位の処理を行うことが多い、1ビット単位で排他的論理和処理の実行も可能である。前者では、少なくとも8回の繰り返し処理(703から711)が必要であり、後者では、少なくとも48回の繰り返し処理(703から711)が必要である。

【0047】正常E転置サブデータを最初に処理するルーチンでは、正常E転置サブデータと暗号サブ鍵の排他的論理和を最初に実行し(802)し、ビット反転E転置サブデータと暗号サブ鍵の排他的論理和を後に実行する(803)。ビット反転E転置サブデータを最初に処理するルーチンでは、実行順序が逆転する(902と903)。

【0048】正常E転置サブデータと暗号サブ鍵の排他的論理和の結果とビット反転E転置サブデータと暗号サブ鍵の排他的論理和は、ビット反転の関係になっている。これは、正常データとビット反転データを同じ暗号サブ鍵と排他的論理和を実行したため、ビット反転の関

係が保存されているためである。

【0049】正常E転置サブデータを最初に処理するルーチン(801)とビット反転E転置サブデータを最初に処理するルーチン(901)は、正常データと暗号鍵データの排他的論理和とビット反転データの排他的論理和を行うため、請求項10の実施例である。

【0050】次に、Sボックスの処理404について述べる。本実施例でのSボックスの処理も基本的な流れは、図24の選択的転置の処理と似た流れである。

【0051】Sボックスの最初に、Sボックスの処理の開始をランダムにするために、乱数を取り出し(1002)、乱数値に従いダミー処理を実行する(1003)。このダミー処理は、ループ処理がないループを乱数値だけ繰り返すような処理でもよい。乱数値に従い、ループの回数が異なるため、Sボックスの処理の開始がランダムになり、データ処理と消費電流の波形の依存性が少なくなる。

【0052】実行フラグと乱数を使った処理順序のランダム化とダミー処理を追加する処理(1004から1013)は、選択的転置処理の2402から2411の処理と同じである。Sボックス処理は、E転置データと暗号鍵との排他的論理和の結果を使って行う。暗号鍵と選択的転置の実行結果の排他的論理和処理(403)の実行結果も図6のような正常データとビット反転データの対の配列からなるデータである。実行インデックスを持つ排他的論理和の実行結果を取り出すときは(1009)、同様に対のデータを取り出す。正常排他的論理和の実行結果を最初に処理するルーチン(1101)とビット反転排他的論理和の実行結果を最初に処理するルーチン(1201)は乱数でランダムに呼ばれる。

【0053】Sボックスの処理は、排他的論理和の実行結果の6ビット毎に、Sボックステーブルのアドレスを計算して(1702)、Sボックスの変換結果である4ビットのデータを求める(1703)。アドレスの計算は、6ビットの第1ビットと第6ビットの2ビットで行番号を求め、第2ビットから第5ビットの4ビットで列番号を求める。

【0054】Sボックステーブルは、8個ある。それぞれのSボックステーブルは、排他的論理和を処理した結果の6ビットずつの変換に使用する。通常の1番目のSボックステーブル(2901)の1列目から4列目、1行目から4行目のデータを図29に示す。本実施例では、図30のようにSボックステーブルをビット反転データも求めるように拡張する。図30での表示範囲は、同じであるが、Sボックステーブルの要素を二つのフィールドに拡張する。第1のフィールド(列の左側のフィールド、例えば3004)は、図29のSボックステーブルの要素のデータを格納し、第2のフィールド(列の右側のフィールド、例えば3005)は、第1のフィールドのデータのビット反転データを格納する。

【0055】正常排他的論理和の実行結果を最初に処理するルーチン(1101)では、まず、正常な排他的論理和の実行結果から、拡張したSボックステーブルを使い、正常Sボックス処理データとビット反転Sボックス処理データを求める(1102)。次に、ビット反転排他的論理和の実行結果から、拡張したSボックステーブルを使い、正常Sボックス処理データとビット反転Sボックス処理データを求める(1103)。ビット反転排他的論理和の実行結果を使って、Sボックス処理をしているため、1103の処理結果は、使用しない。しかし、データ処理と消費電流の関連性を減らすために、ダミー処理として実行する。ビット反転排他的論理和の実行結果を最初に処理するルーチン(1201)では、処理順序が逆になり、同様に、ビット反転排他的論理和の実行結果からの正常Sボックス処理データとビット反転Sボックス処理データは、使用されない。次のP転置処理に渡されるデータは、正常排他的論理和の実行結果を使った正常Sボックス処理データとビット反転Sボックス処理データである。

【0056】排他的論理和の実行結果から拡張したSボックステーブルを使って、正常Sボックス処理データとビット反転Sボックス処理データを求めるには、まず、排他的論理和の実行結果の6ビットからSボックステーブルのアドレスを計算する(1702)。次に、拡張したSボックステーブル(3001)の正常Sボックス処理データを第1のフィールドから求め、ビット反転Sボックス処理データを第2のフィールドから求める(1703)。

【0057】Sボックスの処理での正常Sボックス処理データとビット反転Sボックス処理データは、請求項1の「正常データ及びビット反転データの組み」の実施例の一つである。また、拡張Sボックステーブル(3001)を使って、正常Sボックス処理データとビット反転Sボックス処理データを生成する処理(1703)は、請求項4の「ビット反転データ生成手段」の実施例でもある。ビット反転排他的論理和の実行結果から、正常Sボックス処理データとビット反転Sボックス処理データを求めるSボックステーブルを使いSボックス変換結果を計算する処理(1103と1202)は、正常データではなく、ビット反転データを処理することにより、正常データもビット反転データも同じように処理することにより、データ処理と消費電流の依存性を減らすものである。これは、請求項3の「正常データ命令及びビット反転データ命令処理手段」の実施例の一つである。Sボックスの処理は、6ビット単位のビット単位ではない換字処理であるので、請求項9の実施例である。正常排他的論理和の実行結果を最初に処理するルーチン(1101)あるいは、ビット反転排他的論理和の実行結果を最初に処理するルーチン(1201)は、正常データに対して換字を行い、換字の結果とそのビット反転した結

果を生成し、かつビット反転データに対して換字を行い、換字の結果とそのビット反転した結果を生成するため、請求項 11 の実施例である。

【0058】P 転置処理(405)について図 13 を用い説明する。P 転置処理も選択的転置処理(2401)と同じである。異なっている点は、P 転置行列(図 16 の 1601)を使用していることである。

【0059】S ボックス処理からは、図 6 のデータ形式で、正常 S ボックス処理データとビット反転 S ボックス処理データが渡される。正常 S ボックス処理データを最初に処理するルーチン(1401)では、正常 S ボックス処理データの P 転置を行い(1402)、次にビット反転 S ボックス処理データの P 転置を行う(1403)。ビット反転 S ボックス処理データを最初に処理するルーチン(1502)では、順序が逆になっている。P 転置は、ビットの位置の入れ替えのため、ビット反転 S ボックス処理データの処理結果であるビット反転 P 転置処理データは、正常 S ボックス処理データの処理結果である正常 P 転置処理データのビット反転の関係になっている。1401 と 1501 は、請求項 12 の実施例である。

【0060】以上、f 関数の処理(402 から 405)の実施例を示した。これは、請求項 13 の実施例である。本実施例では、データ処理の順番のランダム化、ダミー処理の追加、正常データやビット反転データを使うことにより、データ処理と IC カードチップが消費する電流との関連性をわかりにくくし、消費電流の波形を観測しても暗号鍵の推定を困難にすることができる。

【0061】次に、RSA のための実施例を説明する。RSA は、素因数分解が困難なことを利用した暗号アルゴリズムである。暗号化及び復号化では、異なった鍵を使うため、非対称暗号アルゴリズムと呼ばれる。暗号化も復号化も以下のべき乗剰余の計算を使用する。

【0062】

$$y = x ** e \bmod n \quad (式 6)$$

$a = b \bmod n$ は、 $a - b$ が n で割り切れることを示す。復号化のときは、 x は、暗号文で、 e は秘密鍵であり、 y が復号化された平文である。べき乗剰余の計算は、乗算剰余を使って計算することができる。IC カードでは、べき乗剰余を高速に計算できるように、乗算乗除を計算できるコ・プロセッサを持っていることがある。乗算剰余は、以下の式である。

$$【0063】 y = a * b \bmod n \quad (式 7)$$

べき乗剰余を乗算剰余を使って計算する方法を図 18 に示す。この計算方法は、例えば、共立出版株式会社岡本栄司著「暗号理論入門」の 94 ページに示されている。秘密鍵である整数 e の 2 進数展開を $e = e_0 e_1 e_2$

..... $e_{(w-1)}$ とする。すなわち、2 進数展開した結果(今後、 e のビットという)は、 e を 2 進数で表したビット列になる。 y の初期値を 1 とし(1802)、 e の

最上位のビットから、 y を n の法で二乗剰余し(1805)、該当ビットの値が“1”であれば、 y に n の法で x を乗算剰余する(1807)。“0”であれば、1807 の処理は、スキップする。1805 から 1807 の処理を、 e の最上位のビットから最下位のビットまで計算する。

【0064】図 18 の処理手順では、 e のビットが“1”であれば、コ・プロセッサが 2 回動き、“0”であれば、1 回しか動かないため、コ・プロセッサの動きを観測すれば、秘密鍵の e を推定することができる。 e を 2 進数展開したビットに関わらず、コ・プロセッサの動作回数を同じにすると、コ・プロセッサの動きを観測するだけでは、秘密鍵を推定することが困難になる。

【0065】図 19 に、コ・プロセッサの動作回数を同じにして、秘密鍵を推定することが困難にしたべき乗剰余の処理手順を示す。本実施例では、秘密鍵 e のビットの値に関わらず、 y に x を乗算剰余する(1906)。その代わり e のビットが“1”であれば、その結果を y にいれる(1908)。そうでなければ、 y をそのまま使用する(1909)。これを e の最下位ビットまで処理を続ける。本実施例では、 e のビットの値に関わらず、コ・プロセッサの動作回数が同じであるため、消費電流波形からコ・プロセッサの動きを観測しても、秘密鍵 e のビットの値は分からない。本実施例は、請求項 14 の実施例である。請求項 14 の乗算剰余処理手段は 1906 として実現されており、乗算剰余処理結果選択処理手段は 1907 から 1909 によって実現されている。

【0066】図 19 に示したべき乗剰余計算を計算する方法より高速な方法として、図 20 に示す方法がある。共立出版株式会社岡本栄司著「暗号理論入門」の 95 ページにて記載されているアディション・チェインという方法を使ったものである。これは、図 19 に示した方法が、 e の 1 ビットずつの処理であったのに対し、複数ビットをまとめて処理する方法である。図 20 に示す方法は、2 ビットをまとめて示す方法である。まず、べき乗算される暗号文 x の 0 乗、1 乗、2 乗、3 乗した結果を求める(2002)。次に、 y を 1 に初期化し(2003)、そして、最後のビットまで処理していなければ(2004)、 y を 2 回二乗剰余を行う。これは、 y の値を 2 ビット分上位の桁にシフトすることに対応する。そして、 e の上位の 2 ビットずつから取り出して、それらが“00”、“01”、“10”、“11”の値にしたがって、2010、2011、2012、2013 の処理を行う。それぞれの処理では、 e の 2 ビットの値によって、べき乗剰余計算の途中結果が格納されている y に $t[0]$ から $t[3]$ までの該当する値が乗算剰余される。例えば、2012 の処理は、 e のビットが“10”の場合で、 y に x の二乗剰余の結果である $t[2]$ を掛ける。その後、最後のビットを処理するまで、2004 から 201

3までの処理を繰り返す。なお、最後にeが1ビットだけ残った場合は、その1ビット分は図18か図19の処理を行う。

【0067】図20では、eの2ビットがどの値でも、コ・プロセッサは、2005で2回、2010から2013のどれか1回を実行するため、合計同じ3回動作する。そのため、コ・プロセッサの動作を観測するだけでは、eのビットを推定することは困難である。なお、図19の処理では、1ビットの処理をするために、2回コ・プロセッサが動いたが、図20の処理では、2ビットを処理するために、3回コ・プロセッサが動作しており、1ビットあたり1.5回コ・プロセッサが動作する。そのため、図20の2ビット方式の方が図19の1ビットの方式に比べて、約25%高速である。ここで、処理時間がコ・プロセッサの動作回数に比例する理由は、図20も図19の処理の大部分がコ・プロセッサによる演算で処理時間を消費しているためである。

【0068】しかし、ステップ2010からステップ2013でのt[0]からt[3]は、eの2ビットごとの処理で毎回同じ値を使用するため、データと消費電流の波形の関連が分かれば、eのビットを推定される恐れがある。

【0069】本発明では、t[0]からt[3]までの値を、べき乗剰余演算中に変更することにより、データと消費電流の波形の関連性を減らすものである。図21を用いて、実施例の一つを説明する。図21と図20の違いは、図20の処理の流れに2105と2106が追加されたことである。図21の実施例は、べき乗剰余を行うためにeを2ビットずつ繰り返し処理中に、ある回数毎に、t[i]までの値を変更する。2105では、繰り返しが一定の回数を処理したかを調べる。そうであれば、2106でt[i]の値を変更する。2105での一定の回数として、eの桁を20で割った値が考えられる。これは、2ビット処理を行っているため、t[i]の値を10回変更することに対応する。また、任意の桁を超えたときにtの値を変更するようにも可能である。t[i]の値の変更方法は、いろいろな方法がある。本発明の本質は、べき乗剰余を行うための繰り返しの中でt[i]の値を変更することである。2106では、t[i]に、剰余演算の法であるnに乱数を掛けて加えることにより、t[i]の値を変更する。これは、以下のように行う。

【0070】

r = 乱数 (式8)

t[i] = t[i] + r * n (式9)

変更されたt[i]は、2112から2115のどれかの処

t[i] = t[i] * v mod n

そして、べき乗剰余が終了したときに、掛けた値のnの法での逆数を掛け合わせた分だけ掛けることにより元に戻す方法である。

【0075】

理で、以下のようにyとnを法として乗算剰余される。

【0071】

y = y * t[i] mod n (式10)

= y * (t[i] + r * n) mod n

= y * t[i] mod n + r * n mod n

= y * t[i] mod n

この処理で、r * nの影響は計算から消えてなくなる。

これは、r * n mod nがゼロであるためである。すなわ

ち、r*nはnで割り切れるためである。t[i]は、t[i]と

10 異なった値のため、ある回数毎に、2112から2115で使われているt[i]が変わるため、データと消費電流の波形の関連性を減らすことができる。本実施例は請求項15の実施例である。実施例15の「ビット値対応入力データべき乗剰余計算処理手段」は2102であり、「ビット対応乗算剰余計算処理手段」は、2108から2115である。「ビット値対応入力データべき乗剰余値変更処理手段」は、2105と2106で実現されている。また、本実施例は、請求項17の実施例でもある。実施例17の「ビット値対応入力データべき乗剰余値に剰余演算の法の整数倍を加えるビット値対応入力データべき乗剰余値変更処理手段」は、2106として実現されている。

【0072】「ビット値対応入力データべき乗剰余値変更処理手段」としての別の実施例として、剰余演算の法に掛ける値として2の任意の数rのべき乗の値を使用する方法がある。2のべき乗を使うことにより、法Nに2のr乗を掛ける処理が容易になる。法Nに2のr乗を掛けることは、法Nをrビット分左にシフトするだけで2のr乗を掛けること等価になる。一般に法Nは100ビットから1000ビットまでの整数であるため、法Nと通常の乱数rの掛け算は、処理時間がかかる。シフト操作は、掛け算に比べてきわめて高速である。また、2のべき乗ではなく、2の8乗のべき乗を使うと、シフトが1ビットシフトではなく、8ビットシフト、すなわち1バイトシフトとなり、ビットのシフトより更に高速に実現できる。1バイト左へのシフトは、データを1バイト分左に置くことに相当し、シフト処理ではなく、データ移動命令で実現できるからである。本実施例は、請求項22の実施例である。

【0073】t[i]の値の変更方法として、別の実施例を図22を用いて説明する。図22での変更方法は、t[i]を変更するときに、以下の式のようにある値を乗算剰余して変更する。

【0074】

(式11)

k = vをべき乗剰余した回数 (式12)

s = u ** k mod n (式13)

y = y * s mod n (式14)

50 ここで、u * v mod n = 1

(式 15)

である。これは、以下の式に示すように、途中で v を掛けた回数だけ u を掛けて、元に戻すという方法である。

【0076】

$$y = y * v^{**k} \bmod n \quad (\text{式 16})$$

$$y = y * u^{**k} \bmod n \quad (\text{式 17})$$

$$= y * v^{**k} * u^{**k} \bmod n$$

$$= y * (v*u)^{**k} \bmod n$$

$$= y \bmod n * (v*u)^{**k} \bmod n$$

$$= y \bmod n * ((v*u) \bmod n)^{**k} \bmod n$$

$$= y \bmod n * 1 \bmod n$$

$$= y \bmod n * 1$$

$$= y \bmod n$$

y は、途中で v を掛けているため、 y が k で v をべき乗剰余された値である。 y は、 y に u を k でべき乗剰余したものを掛けたものであり、式 15 を使うことにより、結果は $y \bmod n$ となる。

【0077】処理の流れを図 22 で説明する。まず、2204 で法 n の剰余演算で v とその逆数である u を求める。そして、2207 で、 v が一定にならないように、 v を変形する。変形の一つ方法は、乱数を求め、その乱数で v をべき乗剰余する。また、一定回数毎に、 v をべき乗する数を増やすことも一つの方法である。その値を $t[i]$ に乗算剰余する (2208) する。そして、 e のビットの処理が終わったならば、 u を使って、 v の操作を無効にする演算を行い、正しい y の値を求める (2218)。 n があらかじめわかっており、かつあらかじめ定められた乱数を使った場合、 u を v を掛けた回数だけ掛けた結果 s をあらかじめ求めておくことができる。その値

$$t[i] = t[i] * v \bmod n$$

$$y = y * t[i] \bmod n$$

$$= y * t[i] * v \bmod n$$

$$y = y * u \bmod n$$

$$= y * t[i] * v * u \bmod n$$

$$= y * t[i] \bmod n$$

2308 で式 20 のように $t[i]$ は変形されており、2314 から 2317 のどれかでは式 21 が実行される。その結果を使い、2318 で式 22 の処理を行い、元の式の結果を導出する。

【0081】さらに効率的に実行するためには、毎回、 v を掛けたことを戻すのではなく、一定回数ごとに v を掛けたことを戻してもよい。その時は、式 19 の u を一定回数 v を掛けたことを戻す値を選び、式 19 を一定回数毎とに計算するようにすればよい。

【0082】図 23 において、 v として $(n+1)/2$ を用い、 u として 2 を用いる。 v と u は、 n を法とする剰余演算で逆数の関係にある。すなわち、 $v * u \bmod n = 1$ である。2307 の v の変形処理として、最初の一定回数の繰り返し処理では、

s を使って、

$$y = y * s \bmod n \quad (\text{式 18})$$

とし、 y の元の値を計算することができる。これは、式 17 の操作と同じである。

【0078】本実施例は、請求項 18 の実施例である。

請求項 18 の「ビット値対応入力データべき乗剰余値に、剰余演算の法において掛けると 1 になる二つの値 v と u の中の一方である v 、あるいは v のべき乗を該当剰余演算の法において掛けることを行うビット値対応入力データべき乗剰余値変更処理手段」は 2204 と 2207、2208 で実現されている。また、本実施例は、請求項 16 の実施例でもある。請求項 16 の「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」の実施例は、2218 である。

【0079】次に、図 23 の処理で、「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」を e のビットをすべて計算した後ではなく、 e のビットの計算途中に行う実施例を示す。図 23 の処理では、2302 から 2317 までの処理は、図 22 の 2202 から 2217 までの処理と同じである。2218 に相当する処理を (2318)、2314 から 2317 のどれかを実行した後に行う。例えば、2308 で $t[i]$ に v を乗算剰余した場合、その操作を毎回元に戻す場合は、 $y = y * u \bmod n$ (式 19) を行えばよい。これは、2314 から 2317 で y に掛けられている $t[i]$ は、元に $t[i]$ の v を掛けたものであるからである。

【0080】

(式 20)

(式 21)

(式 22)

$$v = 1 \quad (\text{式 23})$$

それ以降の繰り返し処理では、

$$v = v * (n+1)/2 \quad (\text{式 24})$$

を用いる。

【0083】そして、2318 で、 v が $(n+1)/2$ であるときの計算を 3 回毎に戻すならば、

$$u = ((2^{**4}*2)^{**4}*2)^{**4*2} \quad (\text{式 25})$$

となり、 2^{**85} となる。式 25 は、図 23 の処理では、2 ビット処理を使っているため、 v を y に掛けるのは 2 ビットごとにしか行われたいためと、直前に 2314 から 2317 で v を掛けていることを元に戻すための式である。そして、2318 では、3 回毎に

$$y = y * 2^{**85} \bmod n \quad (\text{式 26})$$

を計算すれば、 $t[i]$ に $(n+1)/2$ を掛けた操作は元に戻

る。2307で、 v が式24を使って、 $((n+1)/2)**2$ と

$$u = (((2**2)**4*2**2)**4*2**2)**4*2**2$$

となり、 $(2**2)**85 = 2*170$ となる。そして、2318では、3回毎に

$$y = y * 2**170 \text{ mod } n \quad (\text{式 } 28)$$

を計算すれば、 $t[i]$ に $((n+1)/2)**2$ を掛けた操作は元に戻る。式25や式27は、 v として $(n+1)/2$ を使うため、あらかじめ値を計算しておく事ができる。図23では、 e を2ビットずつ処理する方式で述べたが、図21や図22、図23で述べた方式は、 e を1ビットずつ処理する方式や3ビットあるいは4ビットごとに処理する方式にも適用できる。

【0084】図23の2318は、請求項16の「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」を、 e のビットをすべて処理した後まとめて実行するのではなく、途中で実行する実施例である。 $v=(n+1)/2$ と $u=2$ を使った式21から式28の実施例は、請求項19の実施例である。図22及び図23の処理は、請求項20の実施例である。請求項20の「ビット値対応入力データべき乗剰余計算処理手段」は、2202あるいは2302の処理である。請求項20の「ビット値対応入力データべき乗剰余値変更処理手段」は、2207と2208、あるいは2307と2308で実現されている。請求項20の「ビット対応乗算剰余計算処理手段」は2210から2217、あるいは2310から2317で実現されている。請求項20の「入力データべき乗剰余値変更逆変換処理手段」は2218あるいは2318で実現されている。

【0085】図21から図23で述べた実施例は、RSAに基づいた実施例であるが、剰余演算を使用したその他の暗号、たとえばエルガマル暗号にも同様に適用でき

$$y * r = (a * r) \text{ mod } (r * n)$$

となる。すなわち、法 N を r 倍し、右辺の a を r 倍すると、左辺も r 倍となることを示している。法 N のデータと消費電流の関連性を消すためには、法 N を変形することが一つの方法である。式32から、法 N に r を掛けて、かつ右辺の a を r 倍して、剰余演算をし、その結果を r で割れば、もとの y を求めることができることを示している。このようにすることにより、法 N の値と消費電流の関連性を隠すことができる。これは、特に「中国人の剰余定理」（共立出版株式会社岡本栄司著「暗号理論入門」の96ページ）を使ってRSAの計算を行うときに、法 N の素因数 p と q を使った剰余計算において p と q の情報を隠蔽するために有効である。

【0091】図31でその処理手順を示す。まず、法 N を変形するために法 N に掛ける値 r を求める（3102）。次に、法 N を変形するために N を r 倍する（3103）。そして、右辺の値も r 倍し（3104）、剰余演算を行う（3105）。そして、結果を元に戻すために、演算結果を r で割る（3106）。

変形されたならば、式25は、

$$(\text{式 } 27)$$

る。図21から図23に示した「ビット値対応入力データべき乗剰余計算処理手段」、「ビット値対応入力データべき乗剰余値変更処理逆変換処理手段」は、より一般的な剰余演算にも適用できるため、請求項26から請求項33までの実施例でもある。また、請求項15から請求項22までは、共立出版株式会社岡本栄司著「暗号理論入門」の95ページにて記載されているアディクション・チェーンという方法をベースにしたが、通常の1ビットごとの乗算剰余を行うべき乗剰余演算にもでも使うことができる。

【0086】剰余演算でのデータと消費電流波形の関連性は、乗算剰余演算の乗数と被乗数だけではなく、法 N の値にも依存している。法 N の値と消費電流波形の関連性を隠すことが、重要な情報を外部に漏らさないために必要となる。RSAなどの暗号処理では、以下のような計算が頻繁に出てくる。

【0087】

$$y = a * b \text{ mod } n \quad (\text{式 } 28)$$

これを一般化すると、以下の剰余計算となる。

【0088】

$$y = a \text{ mod } n \quad (\text{式 } 29)$$

これは、 a を n で割った余りが y であることを示しており、以下の等式と等価である。

【0089】

$$a = n * x + y \quad (\text{式 } 30)$$

この両辺に任意の数 r を掛けると、以下ようになる。

【0090】

$$a * r = n * x * r + y * r \quad (\text{式 } 31)$$

これを剰余計算の式であらわすと

$$(\text{式 } 32)$$

【0092】図31の3103は、請求項23の剰余演算の法を変更する剰余演算法変更処理手段であり、3104は、剰余演算の右辺の値を変更する剰余演算右辺変更処理手段であり、3106は、剰余演算法変更処理手段で変更した剰余演算の法を変更前の計算結果に戻す剰余演算法変更処理逆変換処理手段の実施例である。図31では、請求項32に示すように剰余演算の法と剰余演算の右辺を変更する手段として、任意の数 r を掛けている。そして、剰余演算法変更処理逆変換処理手段では、 r で割ることを使用している。

【0093】法 N などを変形する r として、2のべき乗を用いると、法 N と右辺の値の r 倍の処理は、べき乗分 N を左にシフトすればよい。そして、剰余演算の結果を r で割る処理はその値を右にシフトするだけでよく、掛け算や割り算が出てこないため、高速に演算ができる。また、2の8乗のべき乗を r として用いると、掛け算や割り算は、シフト命令ではなく、データ移動命令となり、より高速に実現できる。本実施例は、請求項25の

実施例である。

【0094】

【発明の効果】本発明によれば、ICカードチップでの処理手順をランダムにすることや、ダミー処理を挿入すること、処理データを変形することにより、消費電流の波形から、処理や暗号鍵の推測を困難になる。

【図面の簡単な説明】

【図1】 ICカードのハードウェア構成。

【図2】 ICカード用チップ内のハードウェア構成。

【図3】 DESの全体処理の流れ。

【図4】 DESのf関数の処理の流れ。

【図5】 消費電流の波形。

【図6】 正常データとビット反転データのデータ構造。

【図7】 暗号鍵と選択的転置の結果と排他的論理和の処理。

【図8】 正常E転置サブデータを最初に処理するルーチン。

【図9】 ビット反転E転置サブデータを最初に処理するルーチン。

【図10】 Sボックス処理。

【図11】 正常排他的論理和の実行結果を最初に処理するルーチン。

【図12】 ビット反転排他的論理和の実行結果を最初に処理するルーチン。

【図13】 P転置処理。

【図14】 正常Sボックス処理データを最初に処理する

ルーチン。

【図15】 ビット反転Sボックス処理データを最初に処理するルーチン。

【図16】 P転置行列。

【図17】 Sボックステーブルを使いSボックス変換処理するルーチン。

【図18】 RSAの処理ルーチン。

【図19】 暗号鍵のビットの値に関わらず乗算剰余処理を行うように変更したRSAの処理ルーチン。

10 【図20】 2ビット方式のRSAの処理ルーチン。

【図21】 剰余演算の法の整数倍を加えることによりビット値対応入力データべき乗剰余値を変更するRSAの処理ルーチン。

【図22】 ビット値対応入力データべき乗剰余値変更処理逆変換処理を最期に実行するRSAの処理ルーチン。

【図23】 ビット値対応入力データべき乗剰余値変更処理逆変換処理を途中で実行するRSAの処理ルーチン。

【図24】 選択的転置処理。

【図25】 正常入力データを最初に処理するルーチン。

20 【図26】 正常入力データを最初に処理するルーチン

【図27】 選択的転置行列E。

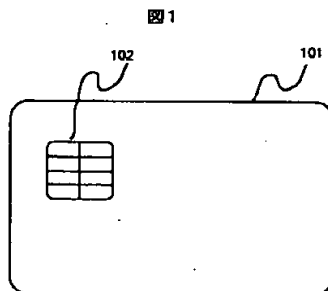
【図28】 アドレスを使ったデータ処理ルーチンの呼び出し。

【図29】 Sボックステーブル。

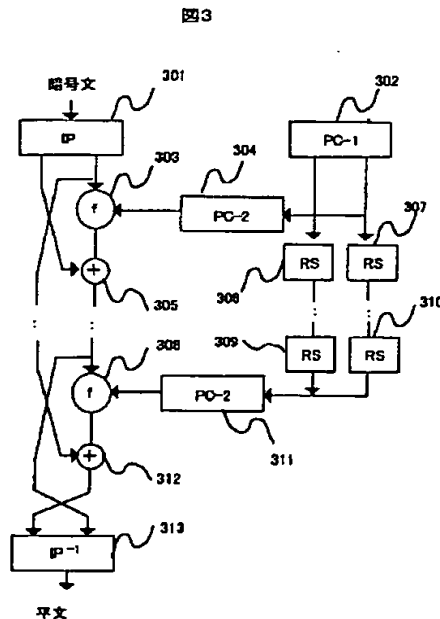
【図30】 拡張したSボックステーブル。

【図31】 剰余演算の法を変形する処理手順。

【図1】



【図3】



【図6】

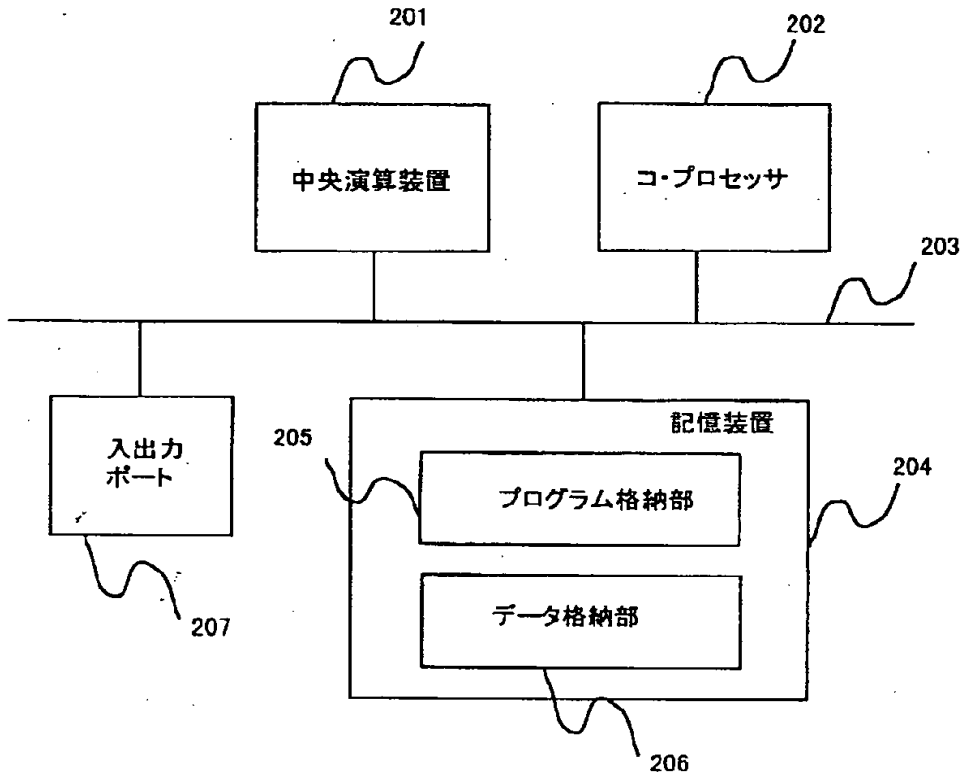
正常データ1	ビット反転データ1
正常データ2	ビット反転データ2
正常データ3	ビット反転データ3
正常データ4	ビット反転データ4

【図16】

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	21	14
32	27	3	9
19	13	30	6
22	11	4	25

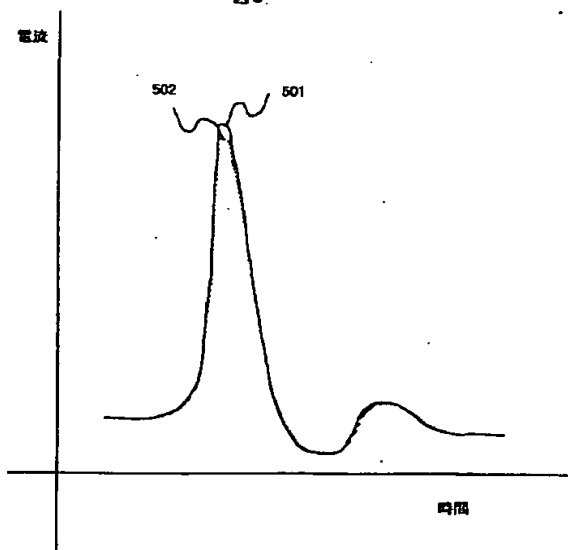
【図 2】

図 2



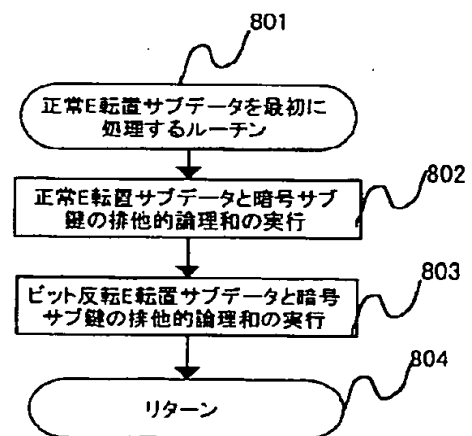
【図 5】

図 5



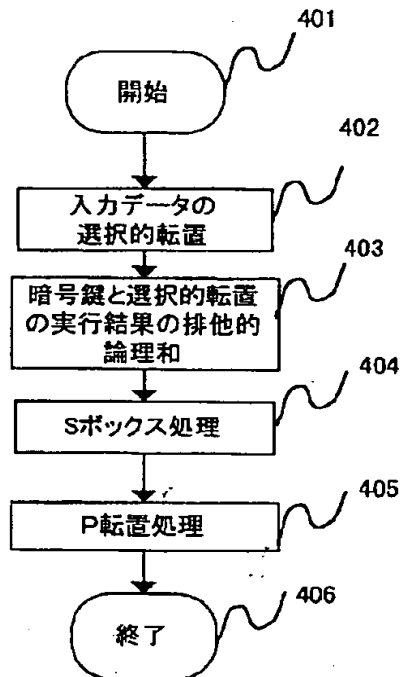
【図 8】

図 8



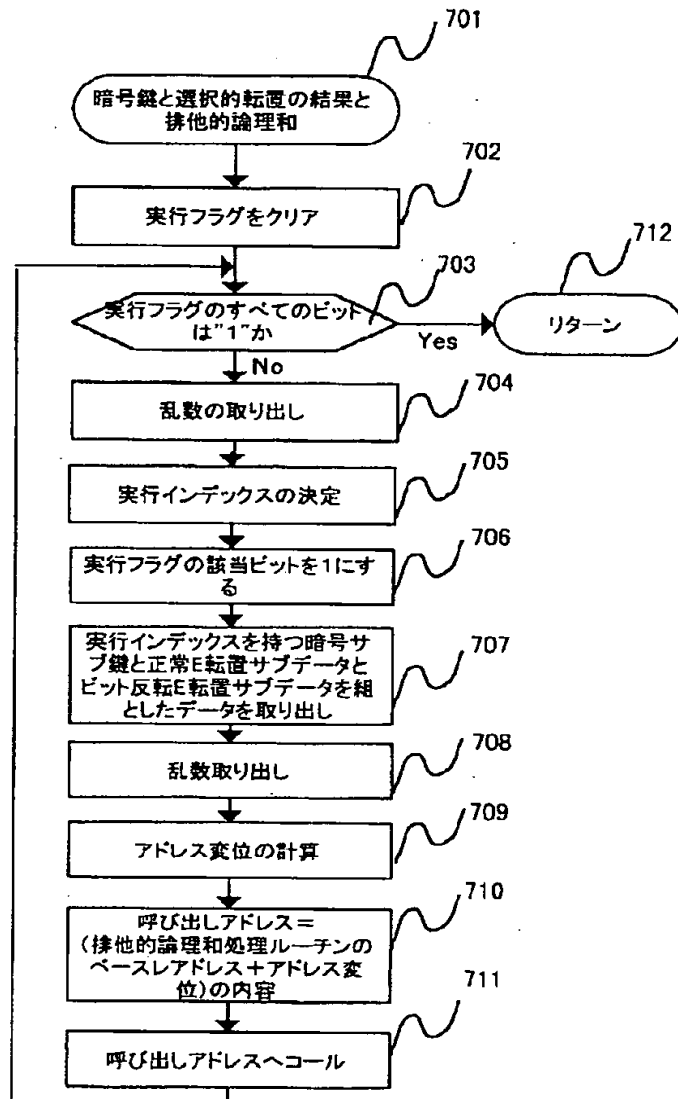
【図4】

図4



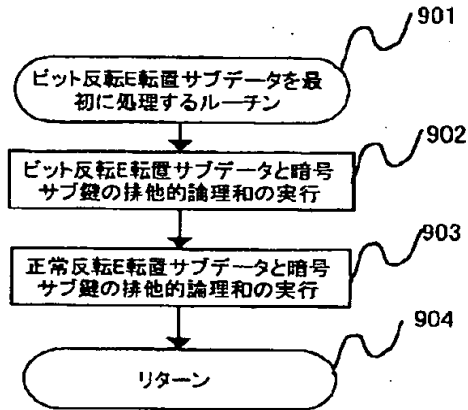
【図7】

図7



【図9】

図9



【図27】

図27

	1	2	3	4	5	6
1	32	1	2	3	4	5
2	4	5	6	7	8	9
3	8	9	10	11	12	13
4	12	13	14	15	16	17
5	16	17	18	19	20	21
6	20	21	22	23	24	25
7	24	25	26	27	28	29
8	28	29	30	31	32	1

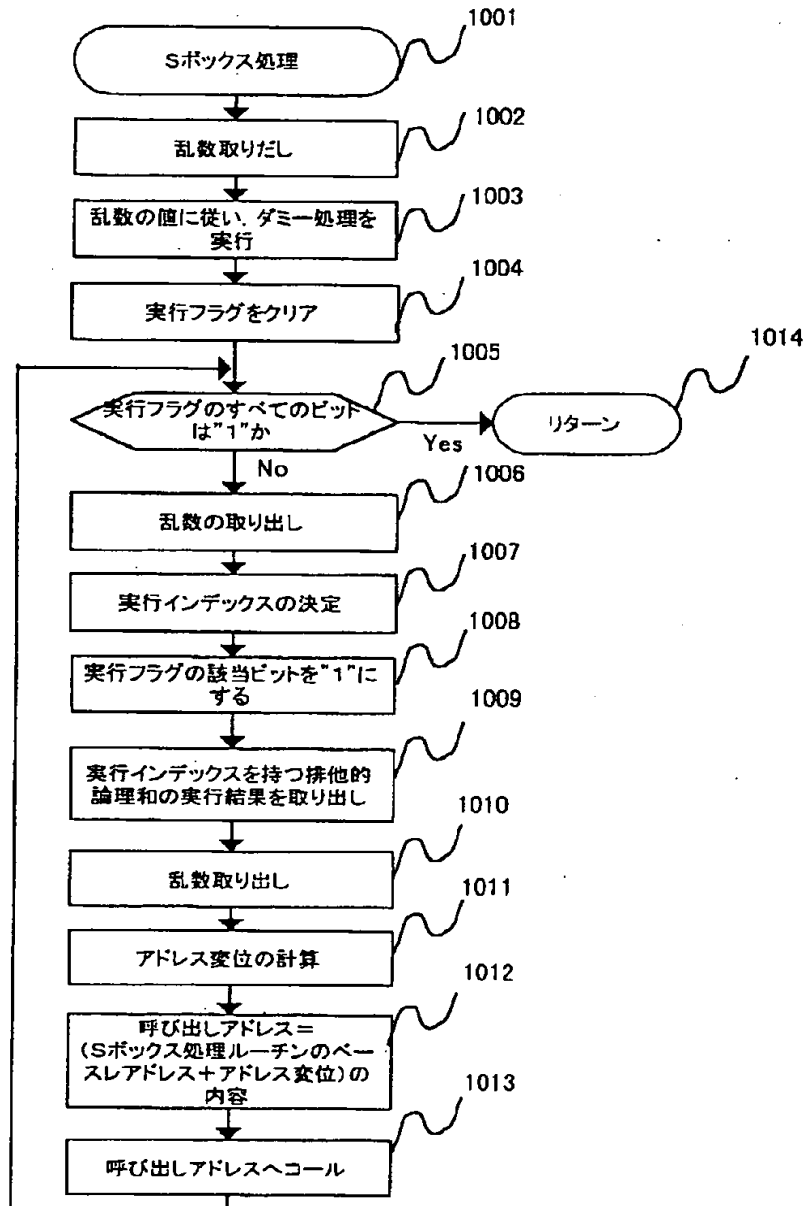
【図29】

図29

	1	2	3	4
1	14	4	13	1
2	0	15	7	4
3	4	1	14	8
4	15	12	8	2

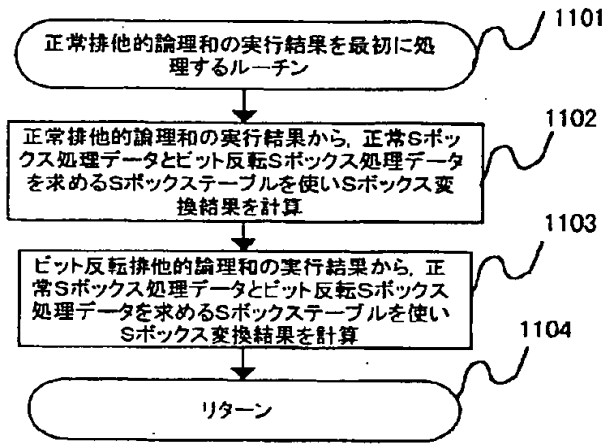
【図10】

図10



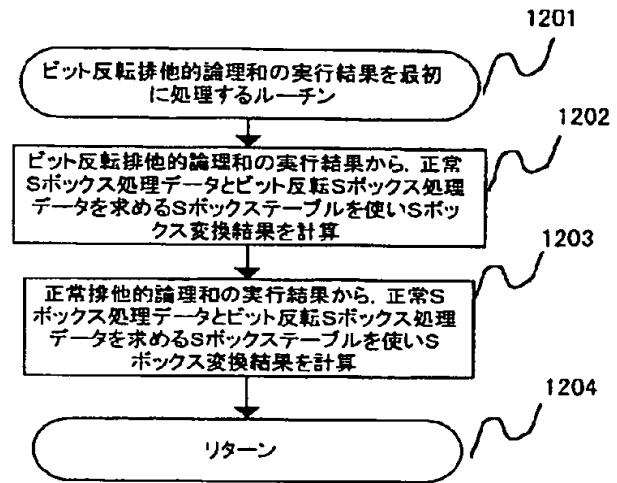
【図11】

図11



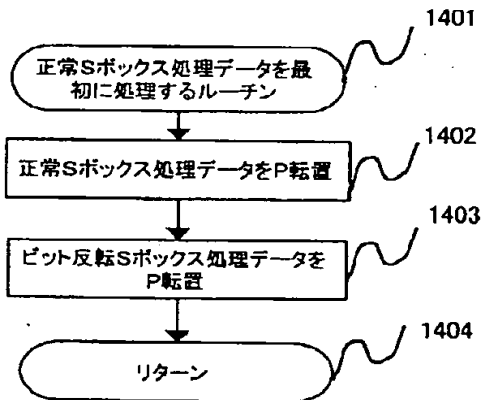
【図12】

図12



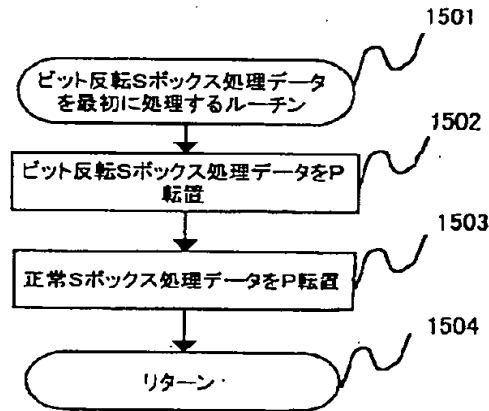
【図14】

図14



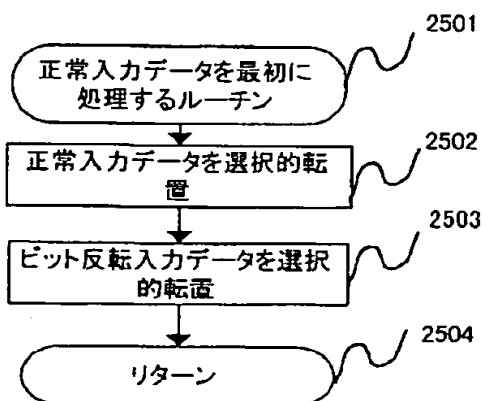
【図15】

図15



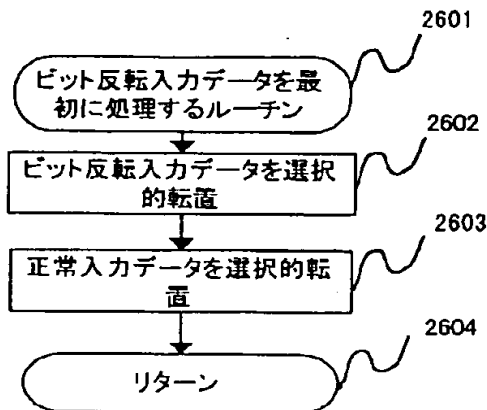
【図25】

図25

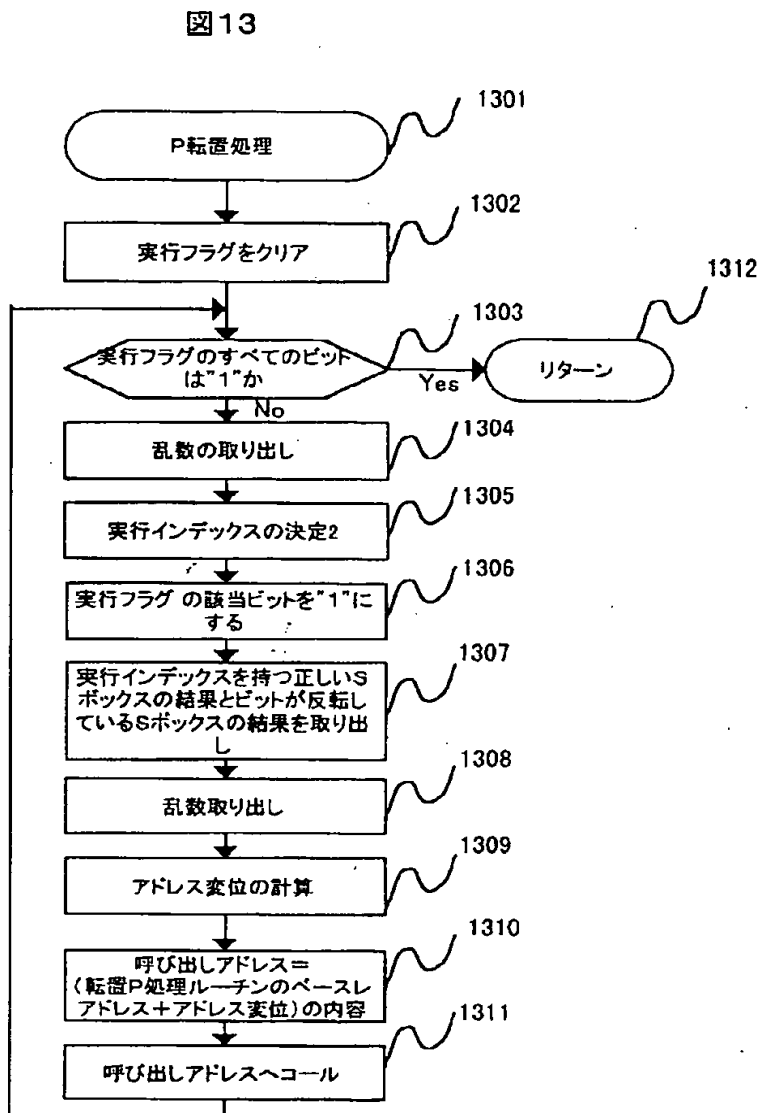


【図26】

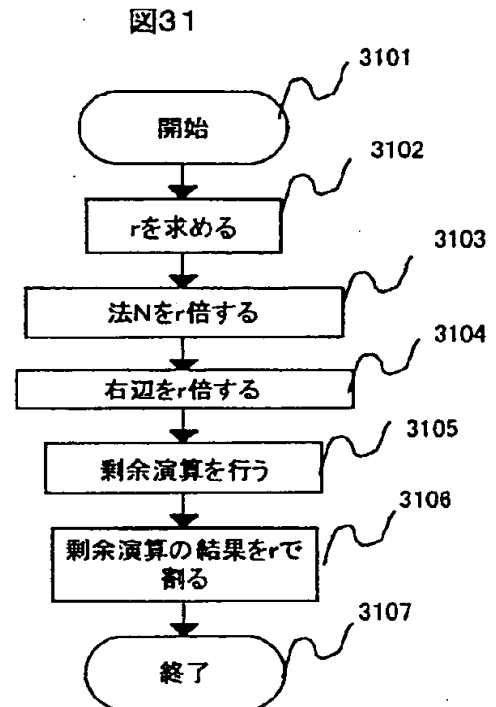
図26



【図13】



【図31】



【図30】

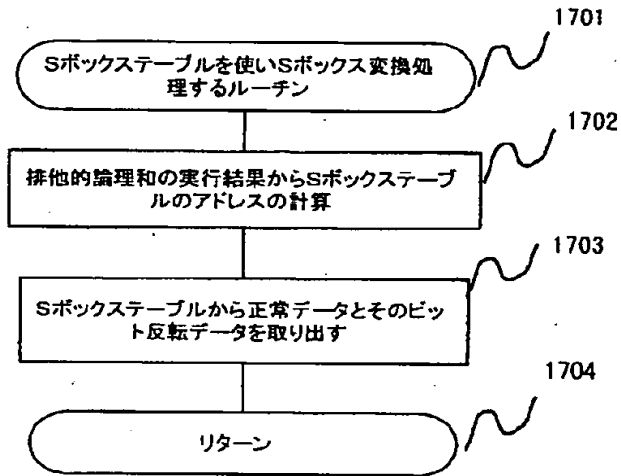
図30

	3001	3004	3005		2	3	4	3003
1	14	1	4	11	13	2	1	14
2	0	15	15	0	7	8	4	10
3	4	11	1	14	14	1	6	7
4	13	0	12	3	8	7	2	13

3002

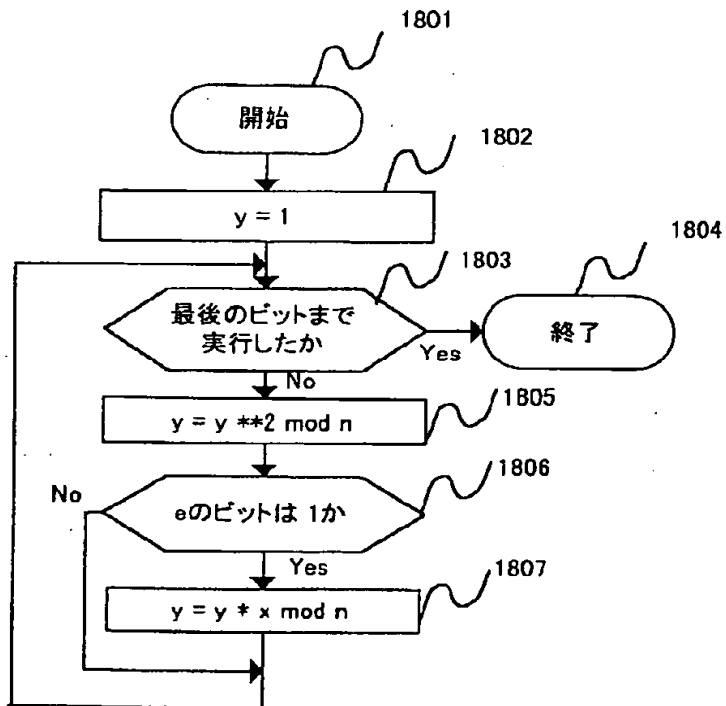
【図 17】

図 17



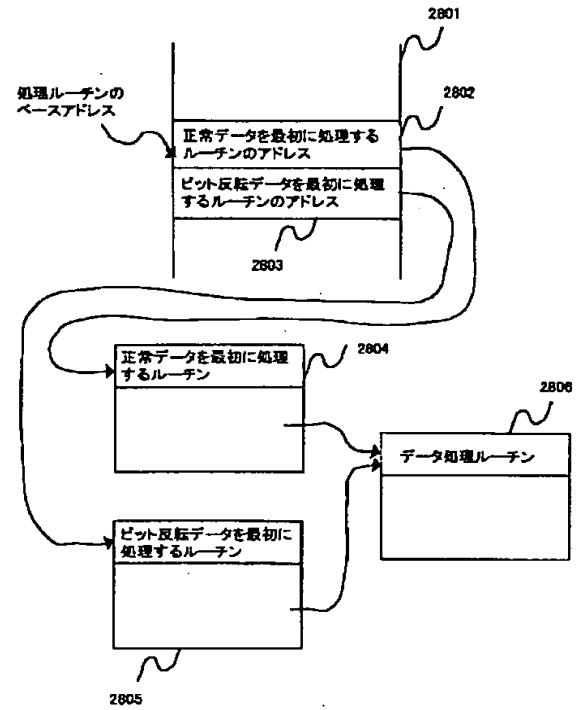
【図 18】

図 18



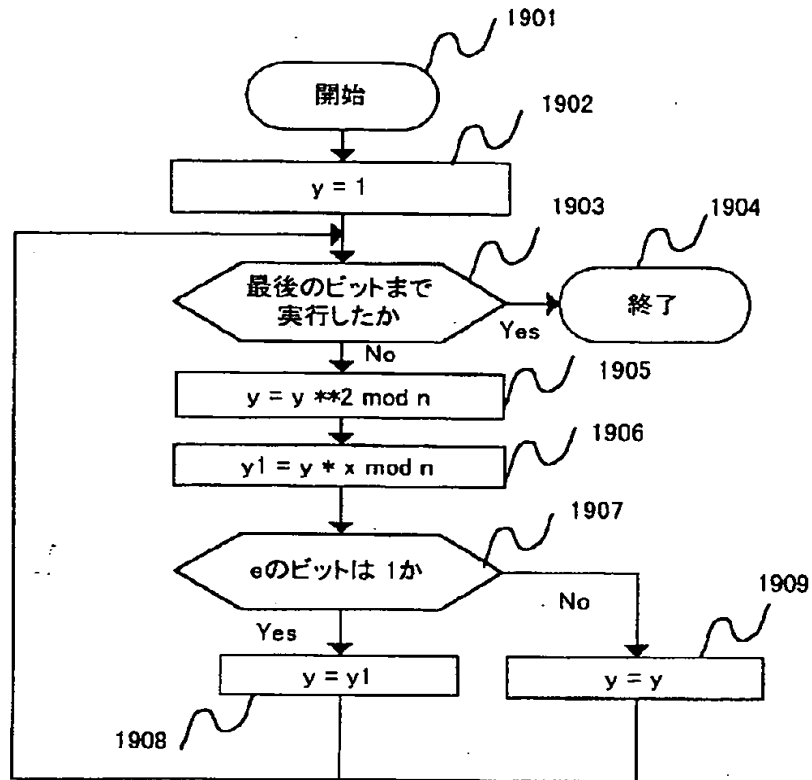
【図 28】

図 28



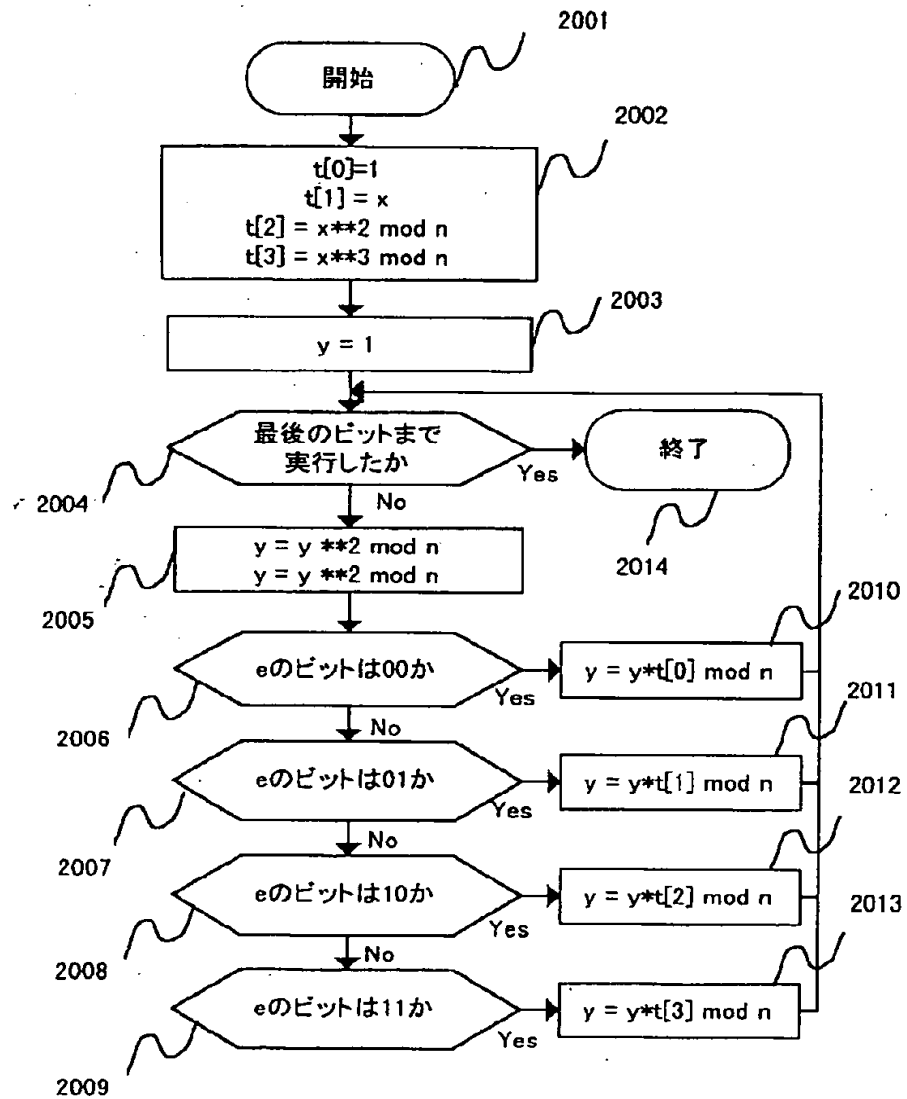
【図19】

図19

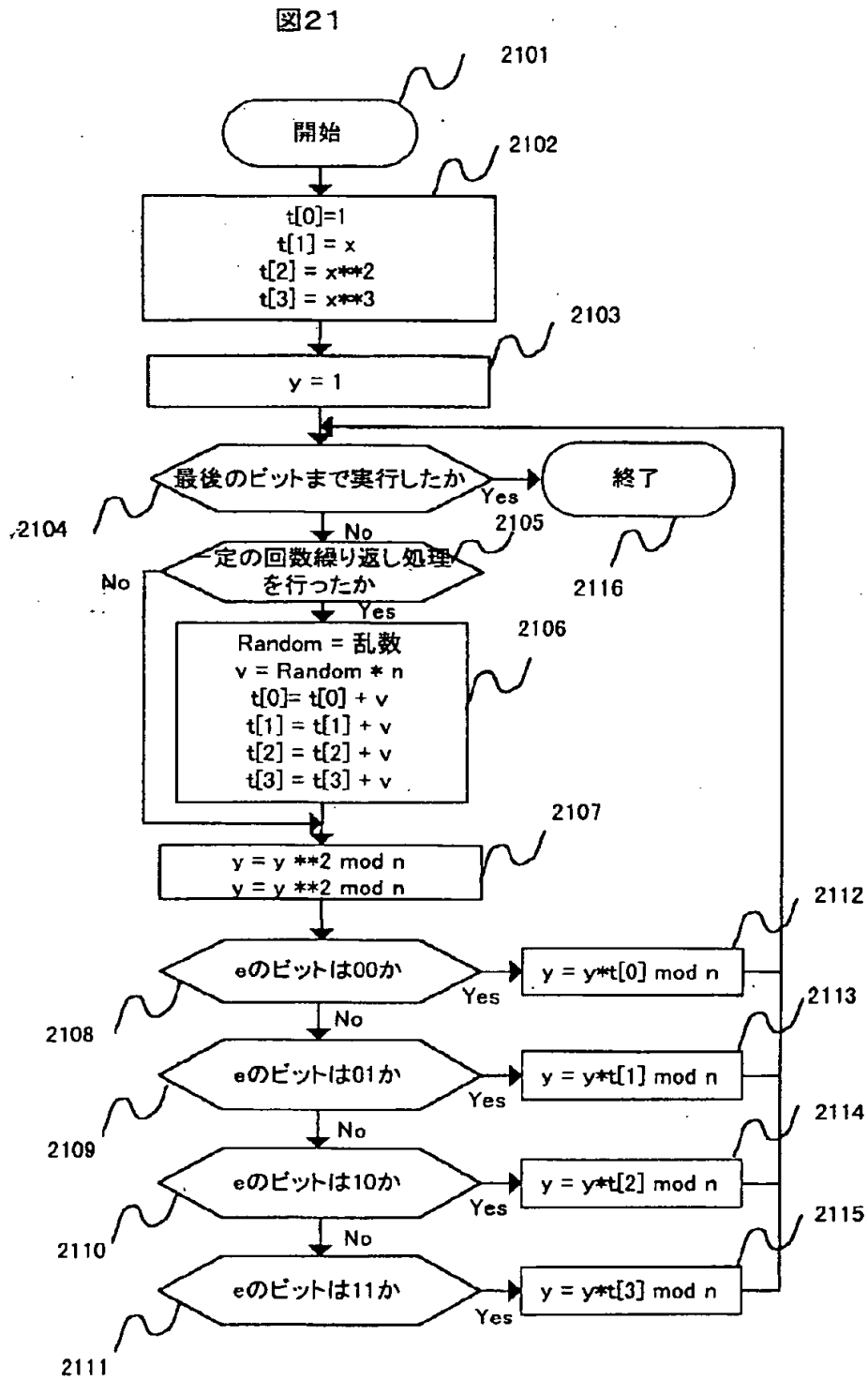


【図20】

図20

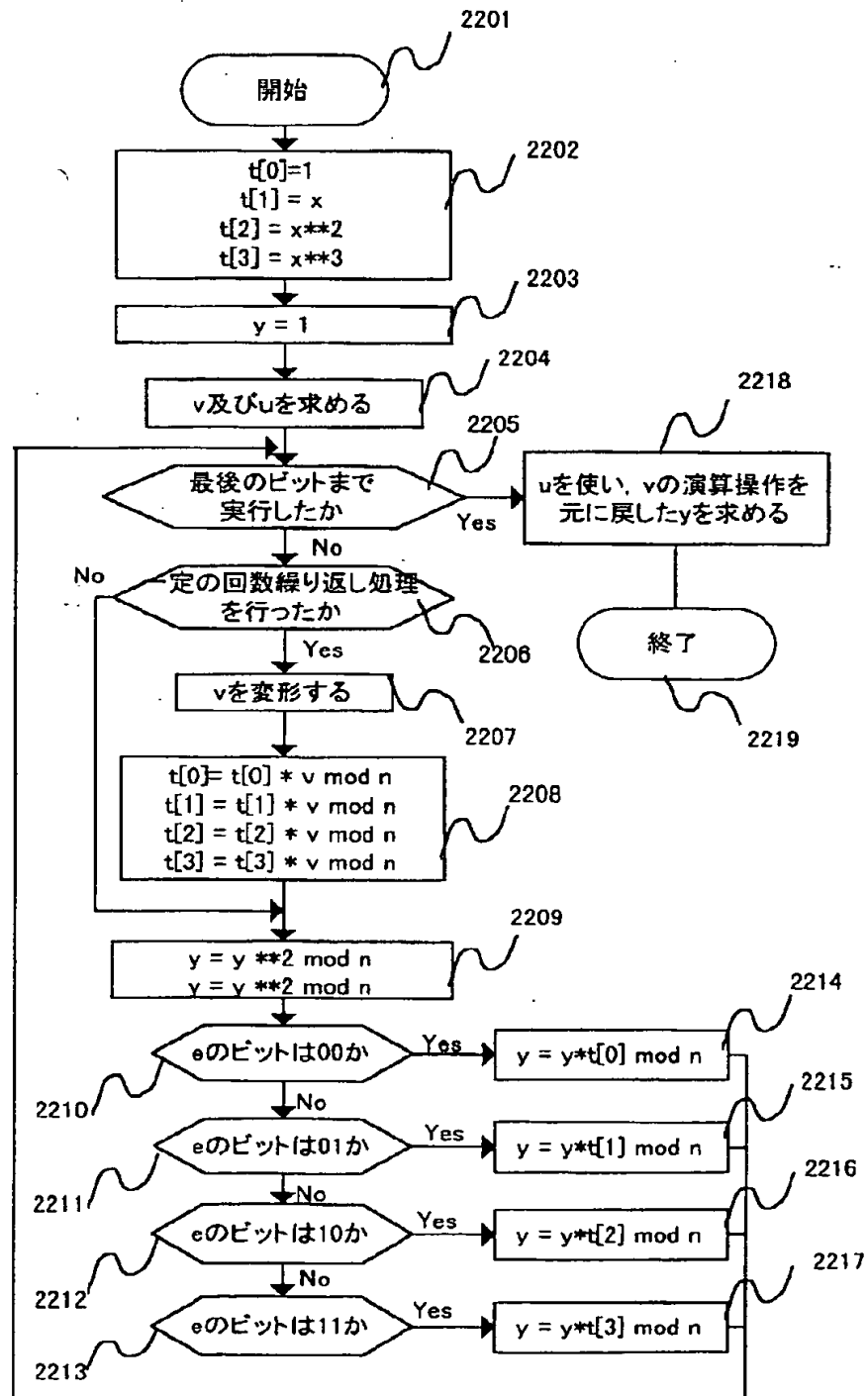


【図 21】



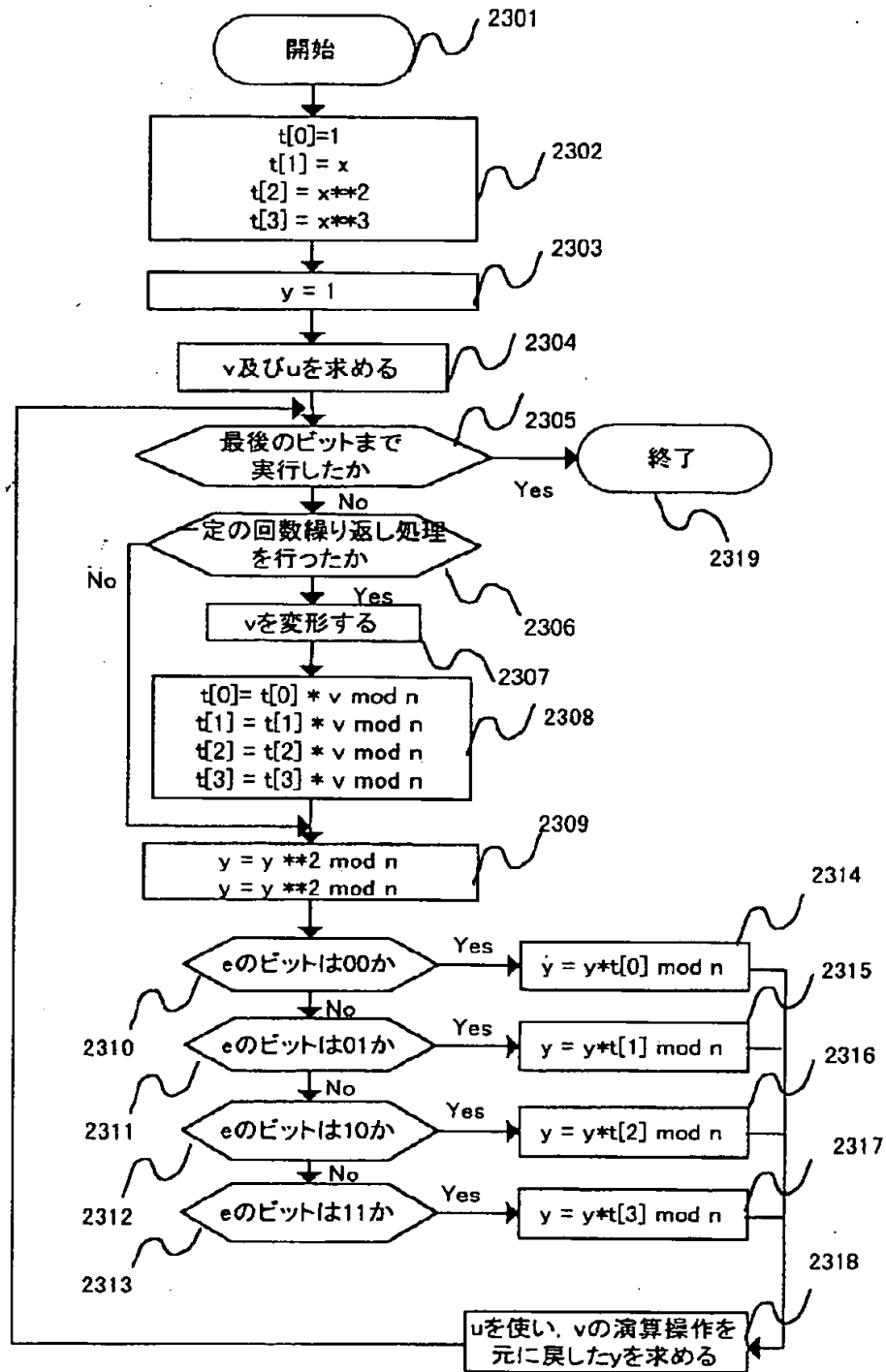
【図 22】

図22



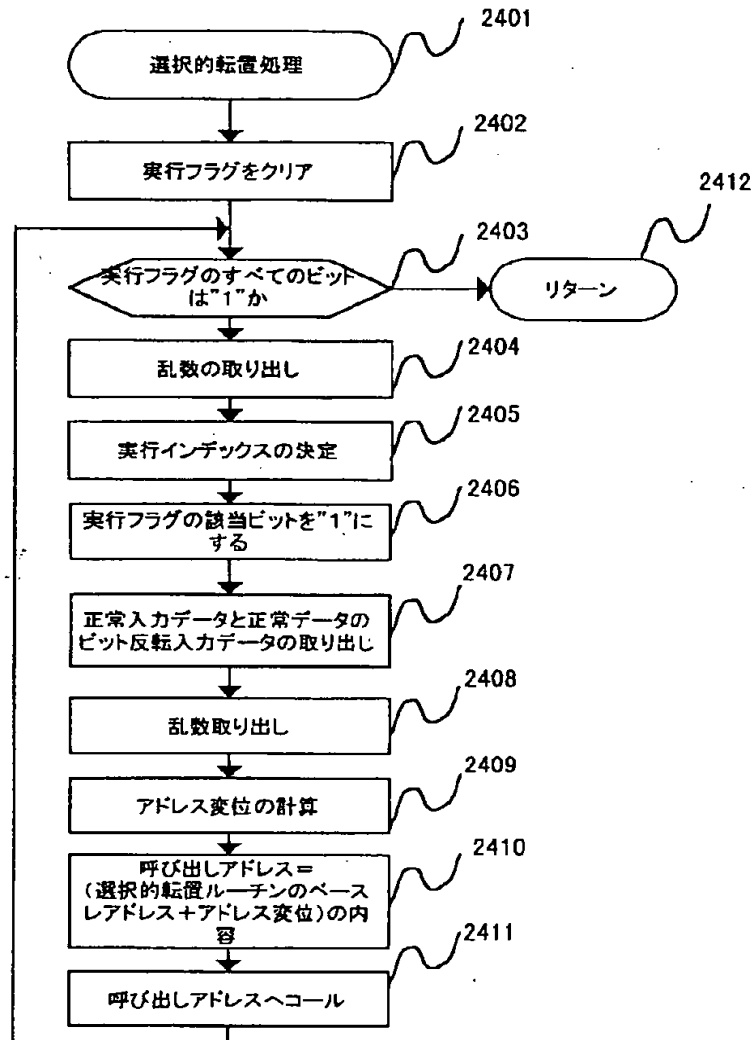
【図23】

図23



【図24】

図24



フロントページの続き

(72)発明者 奥原 進
神奈川県横浜市戸塚区戸塚町5030番地 株
式会社日立製作所ソフトウェア事業部内

(72)発明者 神永 正博
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内
Fターム(参考) 5B035 AA13 BB09 CA29 CA38
5J104 AA41 AA47 EA04 EA08 JA13
JA28 NA08 NA09 NA35 NA40
NA42